オブジェクト指向プログラムを対象とした複雑度メトリクスの実験的評価

Experimental evaluation of the complexity metrics for object-oriented program

神谷 年洋　別府 明　楠本 真二　井上 克郎*　毛利 幸雄†

Summary. Measuring software product and the development process is essential for improving software productivity and quality. Software metrics are quantitative measures of software products and process. For example, complexity metrics is used to evaluate the difficulty of the maintenance of the program. In order to evaluate the complexity of object-oriented program, several complexity metrics have been proposed. Among them, Chidamber and Kemerer's metrics are the most well-known metrics for object-oriented programs. However, they did not clearly describe the way how to apply the metrics to the object-oriented software which are developed by reusing the classes. This paper firstly examines the application method of Chidamber's metrics to the object-oriented program developed by using the framework. Then by using data collected from the actual software development, we experimentally evaluate the effectiveness of the proposed application method.

1　まえがき

近年，ソフトウェアが大規模・複雑化してきたことに伴い，開発期間の短縮やコストの削減・品質向上の要求が高まっている．そのような要求に応えるためには，ソフトウェアの全ライフサイクル (ソフトウェアの開発・保守) にわたる管理が必要である．“ソフトウェア開発プロセスの品質” という概念は，ソフトウェアを開発するプロセスを改善し，そのプロセスで生産されるソフトウェアの品質を安定させ管理可能にするために生まれたものである ?[].

開発プロセスの品質改善のためには，開発プロセスの各フェーズで開発されるプロダクトの状態を測定し，分析して，プロセスにフィードバックする必要がある．ソフトウェアメトリクスは，ソフトウェアのさまざまな特性 (複雑度，信頼性，効率等) を判別する客観的な数学的尺度である．そのなかでも，ソフトウェアの複雑度メトリクスは，ソフトウェアの品質や保守の容易さを評価／予測するために用いられる．測定の結果，ソフトウェアが複雑であればあるほど，エラーが含まれている可能性が高く (低品質である)，保守が困難であると評価される．

これまでに提案された代表的なソフトウェア複雑度メトリクスには，Halstead のメトリクス，McCabe のサイクロマチック数などがある ?[]．Chidamber と Kemerer は，これらのメトリクスは従来の (非オブジェクト指向の) プログラミング言語で開発されたソフトウェアに対する複雑度メトリクスであり，最近のオブジェクト指向設計を用いて開発されたソフトウェアの複雑度を評価するには不十分であると指摘している．そこで，オブジェクト指向設計で開発されたソフトウェアを対象とする

*Toshihiro Kamiya, Akira Beppu, Shinji Kusumoto, Katuro Inoue, 大阪大学

†Yukio Mohri, 日本ユニシス株式会社

6 つの複雑度メトリクスを提案している ?[].

一方，オブジェクト指向設計を用いたソフトウェア開発では，ソフトウェアを独立性が高く，組み合わせの容易な部品を利用してソフトウェアを開発することが目的とされている．すでに存在する高品質のソフトウェアを再利用することで品質の向上を実現し，また，再利用によって開発するソフトウェアの分量を減少させることで開発期間の短縮を目指している．しかし，Chidamber らのメトリクスの評価では，積極的な再利用を用いて作成されたソフトウェアに対しては，有効性の評価が十分に行われていない．

本研究では，Chidamber らのメトリクスを，再利用を積極的に用いて作成されたソフトウェアを対象として適用する方法を検討し，その有効性を評価する．具体的には，まず，教育環境で行なわれたオブジェクト指向言語を用いたソフトウェア開発プロセスから収集したデータを用いて，メトリクスの値と (エラーの難易度で重み付けをした) エラー数との関係を評価する．次に，フレームワークを用いた再利用度の高いソフトウェアに対する Chidamber らのメトリクスの適用方法について検討する．

以降，??. では準備としてオブジェクト指向言語による開発の特徴と Chidamber らのメトリクスを紹介する．??. では研究の目的を明らかにし，実験の概要を説明する．??. では実験データに対する分析を行う．最後に??. では，まとめと今後の課題を示す．

2　オブジェクト指向ソフトウェアメトリクス

2.1　オブジェクト指向言語を用いた開発

従来の開発では，再利用のためにソフトウェア部品を整理分類した “ライブラリ” を使用してきた．ライブラリを用いた開発では，プログラム全体の処理の流れなどの主要な部分は開発者が開発し，ライブラリから必要な部品をもってきて組み合わせる，という形態になる．

一方，オブジェクト指向言語を用いた開発における再利用では，“フレームワーク” と呼ばれる特殊なライブラリが用いられる．フレームワークとは，開発対象となるドメイン固有のソフトウェアアーキテクチャを抽出し，その特徴を構造に反映したクラスライブラリであり，一定の設計思想によって組み立てられたクラスの結合体である ?[]．フレームワークを用いた開発では，プログラム全体の処理の流れなどの主要部分をフレームワークから取り出し，新たに必要な部分を開発者が開発して，それを組み合わせることになる．GUI ドメインは，フレームワークの実用化がもっとも進んでいるドメインであり，MFC(Microsoft Foundation Class) 等の商品化されたフレームワークが存在する ?[]．3. で述べる評価実験でも MFC が用いられる．

2.2　Chidamber らのメトリクス

Chidamber と Kemerer が提案した 6 つのメトリクス ?[] は，オブジェクト指向設計のための複雑度メトリクスであり，クラスの定義からその複雑度を測定する．いずれも数値が大きいほど，より複雑であることになる．

WMC(Weighted Method per Class; クラス当たりの重み付きメソッド数) : クラスのメソッドがどれも同じ程度の複雑度であると仮定できるときは，評価対象のク

ラスのメソッド数となる．

DIT(Depth of Inheritance Tree of a class; 継承木の深さ)：クラスの派生関係が木であると仮定できるときは，評価対象のクラスから根に到るまでのパスの長さ．

NOC(Number Of Children; 子クラスの数)：評価対象のクラスから直接派生しているクラスの数である．

CBO(Coupling Between Object class; クラス間の結合)：評価対象のクラスが“結合”しているクラスの数である．結合とは，あるクラスが他のクラスの属性やメソッドを参照することを意味する．

RFC(Response For a Class; クラスに対する反応)：評価対象のクラスのメソッドの集合と，そのクラスのメソッドが呼び出す他のクラスのメソッドの集合の和集合の要素数である．

LCOM(Lack of Cohesion in Methods; メソッドの凝集の欠如)：評価対象のクラスのメソッドのすべての組み合わせのうち，参照する属性に共通するものがない組み合わせの数から，共通するものがある組み合わせの数を引いたものである．

Chidamber らは 2 つのソフトウェア開発組織でオブジェクト指向言語 (C++と Smalltalk) を用いて開発されたプログラムに含まれるクラスからこれらのメトリクスの値を算出し，クラス毎のメトリクスの値の平均値が大きいほど開発費用が大きくなることを実験的に確かめている ?[]．しかし，メトリクスを計測する際，再利用されるクラスは新規開発されるクラスよりも品質が高いことは考慮されていない．

3 評価実験

3.1 目的

本研究の目的は，大規模な再利用を行うオブジェクト指向プログラム開発プロセスを対象として，再利用を考慮して Chidamber らのメトリクスを適用する方法を提案し，その評価を行うことである．まず，実際のオブジェクト指向プログラム開発プロセスからデータを収集する．収集したデータに対して，Chidamber らのメトリクスの値と，開発プロセス中でエラーを修正するために要した時間との関係を調べる．次に，オブジェクトクラスの再利用を考慮した Chidamber らのメトリクスの適用方法を述べ，この適用方法により予測性が改善されることを示す．

3.2 メトリクスの適用方法

Chidamber らのメトリクスは，再利用されるクラスと新規開発されるクラスを区別せずに扱っている ?[]．しかし，再利用されるクラスは新規開発のクラスよりもエラーが少ない (これらは Basilli ら ?[] によっても観察されている) ので，新規開発のクラスよりも複雑度が低いと思われる．そこで，次のような仮説を立てる：Chidamber らのメトリクスにおいて，再利用されるクラスに対する「結合」や「参照」は複雑度を増大させない．これに従い，Chidamber らの 6 種のメトリクスの定義を，再利用されるクラスの重みを 0 とするように修正する (新規開発のクラスの重みは従来通り 1 とする)．??で述べたメトリクスの中で，他のクラスとの結合や参照を評価するものは DIT, CBO, RFC である．残り 3 つの WMC, NOC, LCOM は一つのクラス内で閉じたメトリクスであるため値は変化しない．ここで，修正したメトリクスをそれぞれ DITo, CBOo, RFCo とし，次のように定義する．

DITo(継承木の深さ)：クラスの派生関係が木であると仮定できるときは，評価対象のクラスから根に到るパスの長さから，パス上にある再利用されたクラスの数を引いた数である．

CBOo(クラス間の結合)：評価対象のクラスが結合しているクラスのうち，開発者が開発したクラスに対する結合の数である．

RFCo(クラスに対する反応)：評価対象のクラスのメソッドの集合と，評価対象のクラスのメソッドが呼び出す他のクラスのメソッドのうち開発者が開発したクラスに属するメソッドの集合との和集合の要素数である．

本論文では，これらの修正版のメトリクスが有効であるかどうかを実験的に評価する．

3.3 実験概要

日本ユニシス株式会社の 1996 年度新人研修における C++プログラム開発演習からデータを収集した．研修生 (被験者) は演習の前に，オブジェクト指向設計，オブジェクト指向言語について講習を受けている．この演習では，6 つのチームが独立に同じ課題を行った．各チームは 4〜5 名の被験者で構成されている．

開発プロセスはウオーターフォールモデルで行われた．すなわち，要求仕様定義，設計，コーディング，レビュー，単体テスト，結合テストのフェーズを経た．課題プログラムはいわゆる酒屋問題 ?[] を拡張したもので，データベースを用いた在庫管理，パスワードによるオペレータ認証，売り上げデータのグラフィカルな表示，売上予測等の機能を持つ．課題が渡された時点で，データベースの構造，入出力ファイルフォーマット，および被験者が開発すべきサブシステム (パスワード管理サブシステム，等) が決定されている．つまり，要求仕様定義フェーズと設計フェーズの一部が終了していることになる．開発期間は 5 日間である．開発されたプログラムは，インストラクタによってテストされ，要求仕様を満たすことが確認される．

プログラミング言語は C++であり，処理系は Microsoft Visual C++である．フレームワークとして Microsoft Foundation Class(MFC) を用いた．MFC を用いることは課題の重要な要件であり，ユーザーインターフェイスとデータベースインターフェイスはすべて MFC のクラスを用いて実装される ?[]．図??に，本実験において，あるチームによって開発されたアプリケーションのクラス階層を示す．図??では，新規開発されたクラスは網掛けで示されている．新規開発のクラスはすべてクラスライブラリのクラスから派生していることがわかる．

被験者はそれぞれ，割り当てられたパーソナルコンピュータ (PC) 上で作業を行う．各 PC およびサーバーは同一のネットワークに接続されている．サーバーは 1 時間おきに被験者の作業ディレクトリをバックアップすることで自動的にプログラムソースファイルを収集する．

発見されたエラーは，レビュー報告書，単体テスト報告書，結合テスト報告書に記入される．それぞれのエラーについて，エラーの修正までの作業を記入する報告書がある．メトリクスの値は，上記の報告書に記載されるエラーが含まれる時点，コードレビュー直前のプログラムソースファイルから算出した．

今回の開発では大規模な再利用が行われている．新規開発部分については，行数でチーム当たり 3000 行程度であり，これには空白行やツールによって生成された

行が含まれる．また，開発されたクラスは，すべてクラスライブラリから派生したものである．一方，再利用した部分については，行数でチーム当たり 10000 行程度である．

4　分析

開発はチームを構成して行われているが，課題プログラムは独立した部分プログラムに分割され，チームのメンバーに割り当てられる．実際に，部分プログラム間に渡るようなエラーはほとんど発見されておらず，各開発者はチームの他のメンバーの開発による影響を受けていない．従って，以降の分析は被験者単位で行っている．

なお，収集されたデータに不備のあった被験者は分析の対象から除いた．結果的には，19 人のデータが分析対象となった．

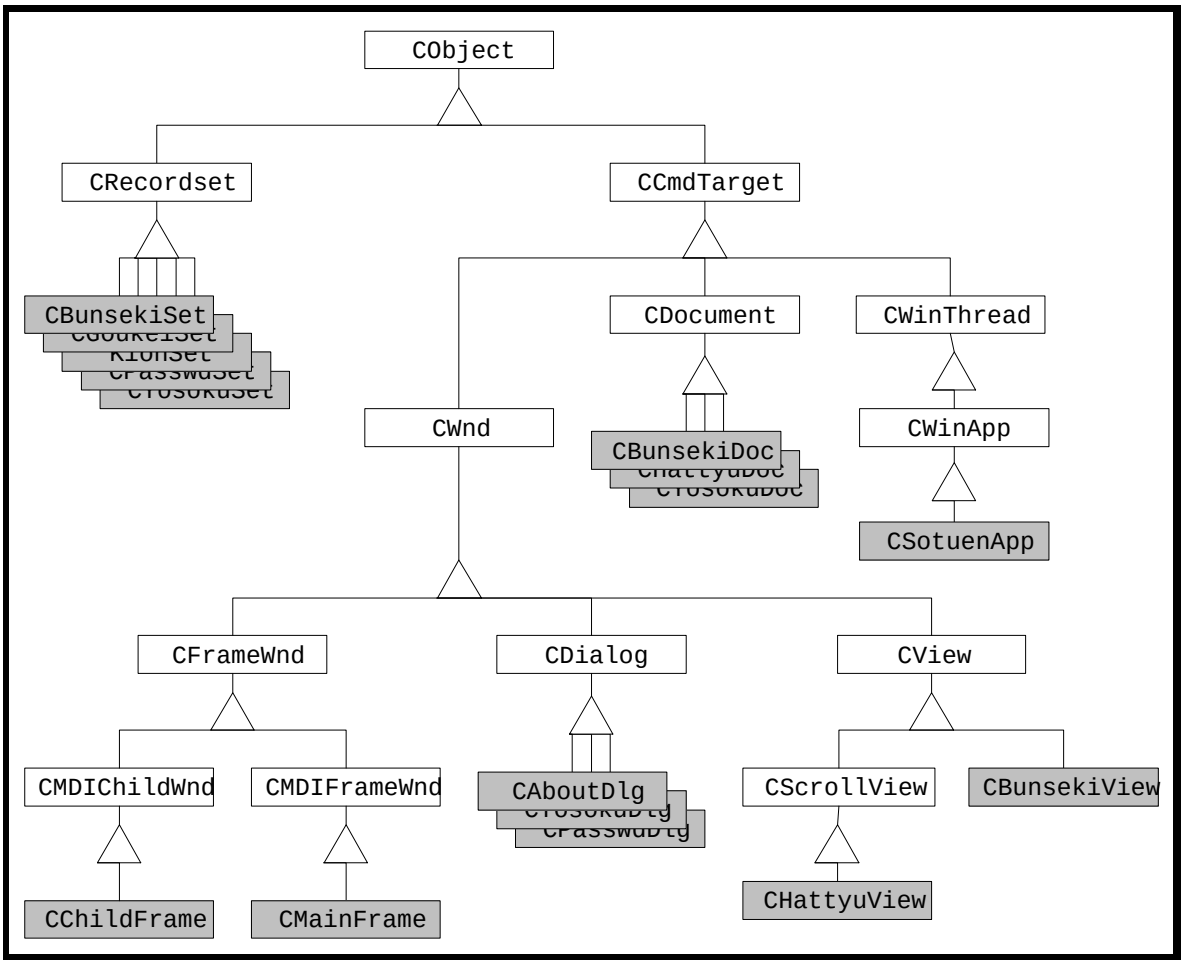

図 1　クラス階層の例

4.1 Chidamber らのメトリクス

表??に，各被験者の開発したプログラムのコードレビュー直前のソースコードについて Chidamber らの 6 種のメトリクスの値を示す．メトリクスの値は各被験者が開発したすべてのクラスについての合計値である．ただし，NOC についてはすべての被験者が開発したプログラムについて，値が 0 であったため省略している．

表??に，メトリクスの値とエラーの数，エラーの修正に要した時間との相関を示す．これらの間に相関があることは有意水準 1%で検定されている ?[]．これより，エラー数とエラーの難易度 (エラーの修正に要した時間) の両方とも相関が高いことがわかる．エラーの難易度については，DIT が 0.77 と高い相関をもっている．一方，RFC は，これらの中では 0.54 と低い値となっている．

4.2 修正版メトリクス

表??に，各被験者の開発したプログラムのコードレビュー直前のソースコードについて CBOo と RFCo の値を示す．表??, 表??より，CBO と CBOo，RFC と RFCo の値にはかなりの差が見られる．CBOo と RFCo とエラーの修正に要した時間との相関を表??に示す．

CBOo と Et の相関は 0.47 であった．一方，CBO と Et の相関は 0.74 であった．CBOo は CBO よりも予測性が低下している．これは，CBO については，フレームワークからのクラスと新規開発のクラスが同じくらい複雑度に寄与していることを意味する．この理由は，相手のクラスの属性を直接参照するような “結合” を行う

表 1　6 種のメトリクスの値とエラー数

被験者	WMC	CBO	RFC	LCOM	DIT	Ec	Et
t1	33	38	75	73	17	7	1124
t2	19	16	38	47	10	2	50
t3	22	21	58	46	14	5	315
t4	7	8	14	16	6	0	0
t5	19	17	41	47	10	2	390
t6	8	8	13	14	6	2	114
t7	19	17	40	47	10	3	21
t8	20	18	32	49	14	7	891
t9	8	9	16	13	6	0	0
t10	25	24	58	62	16	5	530
t11	21	18	52	52	12	8	576
t12	24	20	50	59	16	8	1005
t13	8	9	16	13	6	1	60
t14	38	37	90	88	20	4	850
t15	22	20	55	55	12	3	154
t16	26	23	67	57	16	1	94
t17	11	10	24	11	6	1	90
t18	17	13	24	47	10	3	75
t19	8	9	15	13	6	1	25

表 2　6 種のメトリクスとエラーの相関

	WMC	CBO	RFC	LCOM	DIT
Ec	0.62	0.58	0.54	0.65	0.68
Et	0.72	0.74	0.63	0.7	0.77

場合，結合しているクラスの定義を理解していなければならないため，結合しているクラスが再利用されたクラスであっても，開発者が新規に開発したクラスと同様に，複雑度を増大させるからであると考えられる．

RFCo と Et の相関は 0.77 であった．一方，RFC と Et の相関は 0.63 であった．RFCo は RFC よりも予測性が向上している．これは，RFC については，フレームワークのクラスは新規開発のクラスほど複雑度に寄与していないことを意味する．

結果として，大規模な再利用が行われている場合では，CBO については，再利用された部分も対象にして評価を行う方が，より正確に複雑度を測定できる．一方，RFC については，再利用された部分は対象とせずに評価を行う方がより正確に複雑度を評価できることが確認された．

5 まとめ

本研究では，フレームワークを用いた開発において，クラスの再利用が Chidamber らのメトリクスの CBO と RFC に影響を与えることを明らかにし，その適用方法について検討した．今後の課題としては，より大規模な開発を対象としてメトリクスの評価を行うことや，クラスライブラリの再利用の度合いを考慮に入れた複雑度メトリクスの開発を検討している．

参考文献

[1] V. R. Basili, L. C. Briand, and W. L. Melo: “A Validation of Object-Oriented Design Metrics as Quality Indicators”, IEEE Trans. on Software Eng. Vol. 22, No. 10, pp. 751-761

表 3 CBOo, RFCo の値とエラー数

被験者	CBOo	RFCo	Et
t1	8	31	1124
t2	3	17	50
t3	3	19	315
t4	0	9	0
t5	3	17	390
t6	0	9	114
t7	2	17	21
t8	0	24	891
t9	0	8	0
t10	2	24	530
t11	1	19	576
t12	1	22	1005
t13	0	8	60
t14	3	35	850
t15	3	20	154
t16	3	24	94
t17	0	10	90
t18	0	16	75
t19	0	5	25

表 4 CBOo, RFCo とエラーの相関

	CBOo	RFCo
Et	0.47	0.77

(1996)
[2] S. R. Chidamber and C. F. Kemerer: "A Metrics Suite for Object Oriented Design", IEEE Trans. on Software Eng. Vol. 20, No. 6, pp. 476-493 (1994)
[3] 本位田真一, 青山幹雄, 深澤良彰, 中谷多哉子:"オブジェクト指向分析, 設計", 共立出版 (1995)
[4] S. H. Kan: "Metrics and Models in Software Quality Engineering", Addison-Wesley (1995)
[5] 久米均, 飯塚悦功："回帰分析"，岩波書店 (1987)
[6] 松本健一, 井上克郎, 菊野亨, 鳥居宏次："エラー寿命に基づくプログラマ性能の実験的評価 -大学環境におけるプログラム開発-", 電子情報通信学会論文誌 D, Vol.J71-D, No.10, pp.1959-1965 (1988-10)
[7] 山田茂, 高橋宗雄："ソフトウェアマネジメント入門"，共立出版 (1993)
[8] 山崎利治："共通問題によるプログラム設計技法解説"，情報処理学会誌，25，9，p. 934 (1984)
[9] "Visual C++ブック"，マイクロソフト (1996)