C 言語プログラムソースコードの再利用性測定法とその評価

鷲崎 弘宜 [1]　小池 利和 [2]　波木 理恵子 [3]　田邉 浩之 [3]

1 早稲田大学 〒169-8555 東京都新宿区大久保 3-4-1
2 ヤマハ株式会社 〒430-0904 静岡県浜松市中区中沢町 10-1
3 株式会社オージス総研 〒108-0023 東京都港区芝浦 4-13-23

E-mail:1 washizaki@waseda.jp,2 toshikazu_koike@gmx.yamaha.com, 3{Namiki_Rieko,tanabe_hiroyuki}@ogis-ri.co.jp

あらまし　C 言語プログラムソースコードを対象とし，我々が提案済みの複数の再利用性測定法について，一定規模の再利用実績データを用いて統計的に妥当性を評価した．その結果，幾つかの再利用性測定法がそれぞれ異なる程度を持って実際の再利用性測定に有効であることを確認した．

キーワード　再利用性，測定法，メトリクス，プログラム再利用

Reusability Metrics for Program Source Code written in C Language and their Evaluations

Hironori WASHIZAKI [1]　Toshikazu KOIKE [2]　Rieko NAMIKI [3]　Hiroyuki TANABE [3]

1 Waseda University　3-4-1, Okubo, Shinjuku-ku, Tokyo, 169-8555 Japan
2 Yamaha Corporation　10-1 Nakazawacho, Naka-ku, Hamamatsu-shi, Shizuoka, 430-0904 Japan
3 Ogis-RI Corporation　4-13-23 Shibaura, Minato-ku, Tokyo, 108-0023 Japan
E-mail:1 washizaki@waseda.jp,2 toshikazu_koike@gmx.yamaha.com, 3{Namiki_Rieko,tanabe_hiroyuki}@ogis-ri.co.jp

Abstract　We evaluated statistically validity of a set of reusability metrics for C language program source code from our metrics suite by using actual reuse achievement records. As a result of the evaluation, it is found that some of these metrics are quite useful for measuring actual reusability of program source code with each different effect degree.

Keyword　Reusability, Measure, Metrics, C language, Program reuse

1. はじめに

社会で求められるソフトウェアは量・質・種類の全てについて増大しつつあり，一方で，開発に課せられる要求は厳しさを増すばかりのため（短納期・低コスト），ゼロから個別に作り上げていたのではとても量・質・種類を達成できない状況が存在する[1]．その解決には，既存のソフトウェアの全体や部分を別のソフトウェア開発に繰り返し用いる再利用が重要である．ここで「測定できないものは制御できない」[2]との言葉があるように，再利用のための開発（Development for Reuse）や再利用による開発（Development by Reuse）を系統だって制御可能な形で進めるにあたり，対象の再利用のしやすさを客観的に判断するための測定の方法が不可欠である．さらにその再利用性測定法は，精密な妥当性評価の結果により裏打ちされていない限り，現実のソフトウェア開発において十分な納得感・説得力を持って運用困難である．

そこで本稿では，C 言語プログラムソースコードを対象とし，我々が提案済みの複数の再利用性測定法について，一定規模の実測定値を用いて統計的に妥当性を評価した結果を報告する．

2. 再利用性測定の問題

我々はこれまで，主として組込みシステム制御用途の C 言語プログラムの開発や保守およびその評価を実施する中で，C 言語プログラムの構成要素や全体の内部品質（プロダクトを静的に解析して判明する品質）を測定する枠組みの構築を進めている[3][4]. 同枠組みでは国際規格 ISO9126-1 品質モデルをベースとし，上述の問題意識に基づいて再利用性を加えて扱い，C 言語プログラムソースコードの再利用性を静的に測定するための種々の測定法や，過去のプログラム群の測定結果に基づいて算出された評定基準（いわば閾値）が含まれる．

しかしながら，同枠組みにおける再利用性に関する各測定法の妥当性は，主に定性的評価結果との照らし合わせによるものであり，**[問題 A]** 具体的かつ詳細な再利用実績と照らし合わせて実際に再利用性の高いプ

ログラム要素（あるいは全体）の特徴を反映していることを精密に検証していなかった．さらに同様の理由により，**[問題 B]** 個々の測定法が再利用性に与える影響の度合いの違いを分析困難であった．

これらの問題は，我々の枠組みに限らず，他の多くの再利用性測定の試みについても共通する．これまでにプログラムソースコードを中心として，ソフトウェアの再利用性を測定するための種々の測定法が提案されている[5][6][7][8][9][10][11]. それらの妥当性評価の多くは，一定規模のサンプルサンプルを特定観点（例えば再利用実績[5]や定性的評価[10][11]）に従って優秀な群と劣等な群に分け，両群の測定値の分布傾向をもって妥当性を判定するものである．ただし，これらの方法が評価に扱うデータは，対象プログラムの部分や全体の大まかな再利用頻度や，属人性を排しきれない定性的評価に基づき，評価結果の精度が低い可能性があり[問題 A]の解決に至っていない．[問題 B]についても，人手による定性的分析と重み付けが多数であり（例えば複数専門家の意見を AHP 法で集約するなど），解決に至っていない．

3. 再利用性測定法の精密な妥当性評価

そこで我々は，提案済みの枠組みに含まれる再利用性測定法群について，具体的かつ詳細な再利用実績と照らし合わせることで，妥当性を精密に評価し[問題 A]を解決した．さらにその照らし合わせにあたり，実績を目的変数，各測定法の測定値を説明変数とする重回帰分析を実施することで，各測定法が実績に与えている影響を精密に明らかとし，結果として[問題 B]を解決した．

以降において，我々が提案する再利用性測定法群とその導出過程，再利用実績の種類と収集方法，および評価結果を述べる．

3.1. GQM に基づく再利用性測定法の導出

そもそも対象とする測定法が特定の方針や理論に基づいて体系だって導出されたものでなければ，利用にあたり十分な納得感･説得力を得がたい.そこで我々は，Goal-Question-Metric（GQM）法[13]の適用により，再利用性を精密に明らかとしたいという目標（Goal）から，目標達成のために評価すべき質問（Question）を経て，ソフトウェアの特徴を尺度へと写像する測定法（Metric）へと段階的に結びつけて，最終的に測定法群を得た[3]. その過程はあくまで人手によるものであるが，複数の専門家が十分に時間をかけて目標-質問-測定法の階層を繰り返し検討・修正することにより，一定の納得感・説得力が得られるものとなっていると考えられる．

得られた測定法群を，導出において用いた質問も含めて表 1 に示す．表 1 において，4 つの質問に対応付けられる形で 7 つの測定法が導出されている．いずれも外部への依存関係の複雑さを表すものであり，測定値が小さいほど，対応付けられた質問についてより肯定的に回答し，結果として再利用性が高いと想定されることを意図している．MFn022, MFn042, MFn101 は関数を測定対象とする測定法であり，ディレクトリやシステム全体といった上位の単位で測定したい場合は，合計などにより集約する必要がある．MFl はファイル対象，MMd はディレクトリ対象であることを示し，上位の単位で測定したい場合は MFn と同様に集約する必要がある．

3.2. 再利用実績の収集

再利用の具体的かつ詳細な程度は，単純な再利用回数ではなく，再利用時にまったく修正せずに完全な形で再利用された部分の,流用された全体からの割合（再利用率）をみることで精密に分析可能となる．

そこで我々は，派生開発の文脈において測定可能な再利用率を定義し，再利用実績として採用した．ここで再利用率とは，既存のプロジェクトの成果全体を流

表1: 評価対象の再利用性測定法一覧

質問	測定法（ ID, 名称, 定義 ）		
APIが複雑すぎないか	MMd032	被使用外部ファイルの率	モジュール内の全ソースファイルのうち、モジュール外部から使用されているファイルが占める割合。被使用外部ファイルの種類数 / ソースファイルの数。
	MFl195	モジュール外部から使用される関数の種類数	ファイルが属するモジュールの外部から使用される関数（そのファイル内で定義されているもの）の種類数。複数回使用されていても1と数える。
依存するモジュールは多すぎないか	MMd042	依存するモジュールの数	このモジュールが依存している他のモジュールの数。他のモジュールの関数を使用している場合に、依存していると考える。
責務の分割が適切か	MFl190	パラメータ数が0の関数の率	ファイル内の全関数のうち、引数を持たない関数が占める割合。
関数に影響を与える外部要素は限られているか	MFn022	パラメータの数	引数リスト内に宣言された仮引数の数。
	MFn101	外部変数の読み込み回数	外部変数（Cの場合、外部結合グローバル変数）を、関数が読み込んだ回数。同じ変数を2度読み込んだ場合、2と数える。
	MFn042	呼び出し関数の数	関数内にて他の関数を使用する回数。

用して新たな開発を試みるにあたり，何らかの形で流用した量に対する，無修正でそのまま再利用した量の割合である．この割合が大きいほど，派生開発において派生元のプロジェクト成果を容易に再利用できたこととなる．

例えば図 1 において，再利用対象プロジェクトが A～E 等の単位群から構成され，そのうち，A,B,C の 3 つが新規プロジェクトにおいて再利用されたとする．ここで, A および B については機能/非機能要求の違いへの対応その他の理由で幾らかの修正が施されて流用された場合に，C のみがそのままの形で修正無しに再利用されているため，再利用の実態に即した再利用率を 1/3 と算出する．

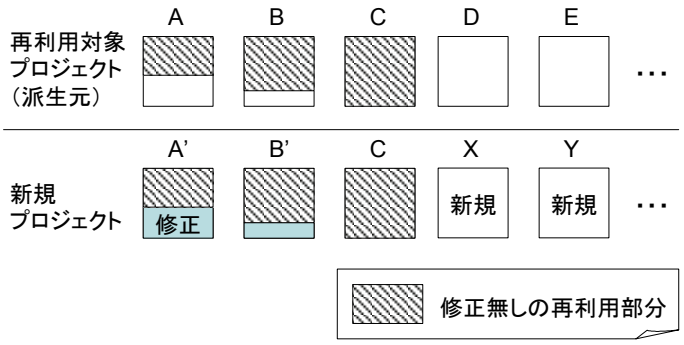

図 1: 再利用率の考え方

上述の考え方に基づいて，我々は再利用の単位として行数，関数，ファイルの 3 種についてそれぞれ対応する形で以下の 3 つの測定法を定義した．

- 再利用行数率：再利用対象プロジェクトのうち全く修正をせずに新規プロジェクトに再利用されたファイルの総行数／新規プロジェクトに流用されたファイルの総行数
- 再利用関数率：再利用対象プロジェクトのうち全く修正をせずに新規プロジェクトに再利用されたファイルの総関数数／新規プロジェクトに流用されたファイルの総関数数
- 再利用ファイル率：再利用対象プロジェクトのうち全く修正をせずに新規プロジェクトに再利用されたファイルの総数／新規プロジェクトに流用されたファイルの総数

続いて，実際の測定値データを選定した．具体的には，実際に 1 度の派生開発が実施され，上述の 3 つの測定値が得られる開発プロジェクト群のうちで，比較的近年のデータで信頼性があり，かつ，統計的に有意な結果を得るためにできるだけ再利用率が異なるように 10 のプロジェクトを選択し,それらを再利用対象プロジェクトとして，上述の再利用率を再利用実績として用いた．

ただし，再利用対象プロジェクトのファイルのうちで，以降の新規プロジェクトにおいて全く再利用されなかった部分も含まれる．プロジェクトデータを表 2 に示す．以降の開発において再利用されなかった部分については，その理由が，対象の動的/静的特徴に基づくものか，あるいは，機能的に必要とされなかったか不明であるため，評価対象からは除外した．

3.3. 妥当性の評価結果

評価の経緯と結果を時間軸に沿って以下に示す．統計的処理の実施にあたり結果に影響を与えるパラメータが多岐にわたるため，下記のように段階を経て注意深く進めた．

表2: 10プロジェクトの規模データ

	ディレクトリ数	ファイル数	有効行数ELOC
総計	213	10,298	5,291,096
被再利用部の合計	173	7,940	3,734,614

(1) 測定単位の選定: 再利用率を目的変数，全ての測定法を説明変数とする重回帰分析を，測定対象の単位を変えて（ファイル，ディレクトリ，システム）全ての再利用率について実施した．実施にあたり目的変数が率データであるため，ロジット変換して用いた．説明変数については生データ，率データ，LOG 変換データの 3 種（後述する）をそれぞれ用いた．その結果，ファイル単位およびディレクトリ単位では全ての再利用率について統計的に有意な結果が得られなかった．この結果は，開発プロジェクトの再利用性に影響する特徴が，そのプロジェクトの単位（すなわちシステム全体の単位）で存在し，ファイルやディレクトリといった要素単位では少なくとも統計的に有意な違いとして現れるような特徴が存在しなかったことを示唆している．つまり，新規プロジェクトにおいて機能追加の必要性などがありえるため，そのような個別の事情に左右される要素の単位ではなく，システム全体の単位で再利用率を測定することが適切と考えられる．

(2) 再利用率および測定値データ形式の選定: 測定単位をシステム単位に固定した上で，再利用率および測定値のデータ形式の全ての組み合わせについて重回帰分析を実施した．測定値のデータ形式として，生データ，適切な母数で正規化した率データ，生データの変動の粗密を均した LOG 変換後の LOG データの 3 種が考えられる．分析の結果，測定値のデータ形式を率データとして，再利用関数率との組み合わせが最も自由度調整済み寄与率が

高く，従って有効であることが判明した．これは，生データがプロジェクトの規模の影響を強く受けるのに対し，規模により正規化した率データが適切であるためと考えられる．

(3) 説明変数の選択: ここまでの分析では 7 つの測定法測定値を全て説明変数として用いてきたが，データ数が 10 個と説明変数の個数に対して不足しており（説明変数の個数の 2 倍はデータ数があることが望ましい)，重回帰分析によって得られた回帰式の信頼性が低い可能性がある．最初に全ての説明変数 7 個を採用した場合には，自由度調整済み寄与率 0.812 となり，7 個中 5 個の説明変数について偏回帰係数の符号が「+」の回帰式を得た．前述のように，表 1 に示した採用した 7 種の測定法は全て測定値が小さいほど再利用性が高いと想定したうえで導出しており，この分析結果はその想定に完全に逆らうものとなっている．そこで統計解析ツールを用いて対話的に説明変数を取捨選択し，自由度調整済み寄与率が最大となる組み合わせを選択した．その結果，MMd042，MMd032，MFn042 を選ぶ場合に，自由度調整済み寄与率が 0.827 と最高の回帰式が得られることが分かった．分析における基本統計量と変数あたりの統計量をそれぞれ表 3，表 4 に示す．表 3 において分散比が大きく危険率 1%で有意の結果を得ており，得られた回帰式が有効であることが分かる．また表 4 において，MMd032（の率データ）および MFn042（の率データ）について偏回帰係数の符号が「－」であり，採用した測定法の測定値が小さいほど再利用性が高いという当初の想定に合致している．なお，MMd042（の率データ）については符号が「+」になっているが，これは完全には独立していないと予想される 3 つの説明変数を用いて重回帰分析を実施した影響と考えられ，同説明変数について標準偏回帰の値が最小で目的変数への影響が最小であるため，大きな問題とはならないと考えられる．得られた回帰式による予測値と実測値の散布図を図 1 に示す．図 1 において相関係数は 0.941 と非常に大きく，散布状況からも得られた回帰式によるシステム単位での関数再利用率の予測値が概ね実測値に近いことが分かる．

表3: 説明変数を選択後の基本統計量

要因	平方和	自由度	分散	分散比	検定
回帰	10.713	3	3.571	15.34	危険率1%で有意
残差	1.397	6	0.233		
計	12.11	9			

表4: 説明変数を選択後の変数あたりの統計量

変数名	偏回帰係数	標準誤差	t値	標準偏回帰	トレランス
定数項	3.446	1.769	1.948		
MMd042率	0.362	0.207	1.744	0.284	0.723
MMd032率	−8.453	2.31	−3.66	−0.687	0.546
MFn042率	−1.029	0.48	−2.14	−0.386	0.594

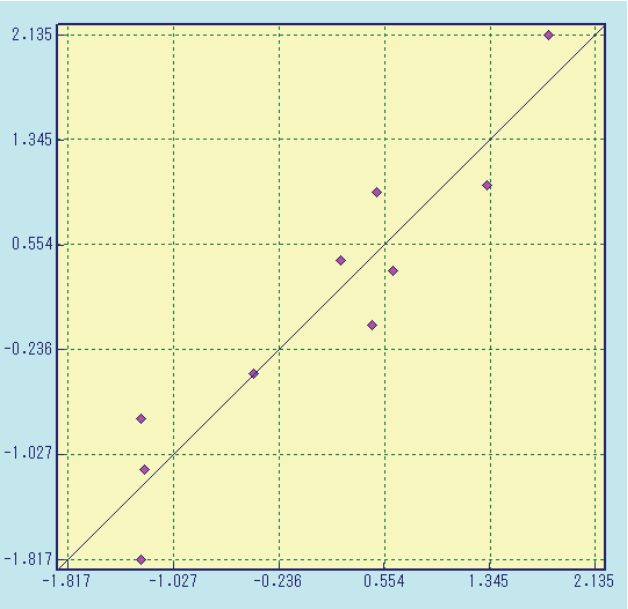

図 1: 回帰式による 10 プロジェクトのシステム単位の関数再利用率のロジット変換後の値の予測値（X 軸）と実測値（Y 軸）の散布図，および，相関グラフ（相関係数=0941）

3.4. 評価結果のまとめ

上述の妥当性評価結果を以下にまとめる．

- 再利用率に基づく再利用実績としては，再利用関数率が再利用性測定結果と最も関係が強い．これは，C 言語プログラムにおける再利用の単位がもっぱら関数であることが原因と考えられる．
- 再利用性を測定する単位としては，要素よりもシステム全体が再利用率と最も関係が強い．これは，開発プロジェクトの再利用性に影響する特徴や傾向が，そのプロジェクトの単位で存在することが原因と考えられる．逆に，システム中の個々のファイルやディレクトリの単位では，そのような特徴や傾向が必ずしも存在しないと考えられる（システム中の特定のファイルやディレクトリにそのような特徴や傾向が偏っている，など)．
- 重回帰分析により，含まれるディレクトリ単位で被使用外部ファイルの率が小さく，関数単位で呼び出し関数の数が少ないほど，システム単位の関数再利用率が高い傾向にあることが分かった．特に，被使用外部ファイルの率の影響が大きいこと

が分かった．ただし，ディレクトリ単位の依存するモジュールの数について再利用性に与える影響については不明な点が残り，その調査は今後の課題である．

以上の評価は，上述のように複数回の回帰分析を経て注意深く実施され，目的変数としての再利用実績としても 3 種の詳細な値を用意して分析したため，提案する再利用性測定法を精密に妥当性評価できたと考えられ，従って[問題 A]を解決した．また，3 つの再利用性測定法のそれぞれの再利用実績への影響の度合いを分析により合わせて明らかとし，従って[問題 B]を解決した．

4. おわりに

C 言語プログラムソースコードを対象として，複数の再利用実績データを用いた精密な分析を通じて，我々が提案済みの複数の再利用性測定法のうちで 3 つがそれぞれ異なる影響の度合いを持って実際の再利用性の大きさに関係している傾向にあることを統計的に明らかとした．その分析にあたり，再利用回数ではなく，現実の再利用の実態を反映可能な再利用率を新たに定義して用いた．本稿の主要な貢献は，再利用性測定法の精密な妥当性評価および個々の再利用性への影響度合いを分析する手順を示したこと，および，その結果として妥当性が評価済みで C 言語プログラムソースコードに適用可能な測定法の集合を，個々の再利用性への影響の度合いを伴って具体的に提案したことである．

今後の課題として以下が挙げられる。

- 再利用実績データとして用いるプロジェクトデータの増加と分析の繰り返しに基づくさらなる測定法の見直し（特に MMd042）と一般性の分析
- 妥当性評価時の照らし合わせ先として用いる再利用実績としての再利用率の定義を派生元と派生先が 1 対 1 に対応していない状況にまで拡大させる（例えばあるプロジェクト成果が複数の新規プロジェクト群によって様々に再利用されている場合の再利用率）
- 再利用回数といった他の再利用実績データとの関係の分析
- 再利用性測定法導出時の質問は特定のプログラム言語に対して非依存であることも考慮したうえで，妥当性が評価された測定法の他のプログラム言語ソースコードへの適用可能性の検討

文　献

[1] 鷲崎弘宜, “大規模ソフトウェアの効率的開発技術を追求 ～再利用と品質保証が鍵～,” 情報通信ジャーナル, 2007 年 5 月号, 2007.

[2] T. DeMarco, “Controlling Software Projects: Management, Measurement & Estimation,” Yourdon Press, 1982.

[3] 鷲崎弘宜, 波木理恵子, 福岡呂之, 原田洋子, 渡辺博之, “プログラムソースコードのための実用的な品質評価枠組み”, 情報処理学会論文誌, Vol.48, No.8, pp.2637-2650, 2007.

[4] H. Washizaki, R. Namiki, T. Fukuoka, Y. Harada and H. Watanabe, "A Framework for Measuring and Evaluating Program Source Code Quality", Proc. PROFES 2007, LNCS, Vol. 4589, pp.284-299, 2007.

[5] 菅野文友, 吉澤正監修, “21 世紀へのソフトウェア品質保証技術,” 日科技連出版社, 1994.

[6] G. Sindre et al., “The REBOOT Approach to Software Reuse,” Journal of Systems and Software, Vol.30, No.3, 1995.

[7] W. Frakes and C. Terry, “Software Reuse: Metrics and Models,” ACM Computing Surveys, Vol.28, No.2, 1996.

[8] L.H. Etzkorn, et al, “Automated Reusability Quality Analysis of OO Legacy Software,” Information and Software Technology, Vol.43, 2001.

[9] 中島哲, 直田繁樹, 堀田勇次, “C++を対象とした再利用に関するメトリクスの測定,” オブジェクト指向シンポジウム'98, 1998.

[10] H. Washizaki et al., “A Metrics Suite for Measuring Reusability of Software Components,” Proc. IEEE METRICS’03, 2003.

[11] 平山雅之, 佐藤誠, “ソフトウェアコンポーネントの利用性評価,” 情報処理学会論文誌, Vol.45, No.6, 2004.

[12] K. Inoue et al., “Ranking Significance of Software Components Based on Use Relations,” IEEE Transactions on Software Engineering, Vol.31, No.3, pp213-225, 2005.

[13] V.R. Basili and D.M. Weiss, “A Methodology for Collecting Valid Software Engineering Data,” IEEE Transactions on Software Engineering, Vol.10, No.6, 1984.