組込みソフトウェアの派生開発におけるソースコードメトリクスによる再利用性測定

Measuring Reusability by Program Source Code Metrics in Derivative Embedded Software Development

早稲田大学 グローバルソフトウェアエンジニアリング研究所
Waseda University Global Software Engineering Laboratory
株式会社小松製作所 開発本部 ICT 開発センタ
Komatsu Ltd. Development Division, ICT Development Center

○鷲崎 弘宜[1)] 森田 翔[1)] 長井 恭兵[1)] 布谷 貞夫[2)] 佐藤 雅宏[2)] 杉村 俊輔[2)] 関 洋平[2)]
○Hironori Washizaki[1)] Sho Morita[1)] Kyohei Nagai[1)] Sadao Nunotani[2)] Masahiro Sato[2)]
Shunsuke Sugimura[2)] Yohei Seki[2)]

Abstract Low understandability and changeability of embedded software could decrease the productivity and quality in later derivative developments due to long time of trying to understand and a broad range of modifications. The effects of static characteristics of source software on reuse have not been adequately discovered although it is preferable to evaluate ease of reuse without significant modifications in derivative developments. In this paper, we propose a framework for measuring reusability of C program source code by specifying static characteristics related to ease of black-box reuse without any modifications and analyzing relations to actual reuse rate in derivative developments. By applying the framework to 10 embedded software derivative development results in a construction equipment manufacturer, we obtained a set of reusability metrics mainly related to module couplings via external global variables; it is confirmed that these metrics could predict the actual reuse rate with high accuracy. As the actual reuse rate, we used the rate of the number of functions reused without any modification in total of the number of functions reused with or without any modification. This achievement is expected to be utilized for predicting future reuse rate and related efforts in derivative developments, and for supporting modification and/or refactoring of software under consideration of derivative development target. In the future, we will verify the generalizability of the framework and findings of the paper. Moreover we have a plan to consider the mechanism of proposing ways for improving reusability according to the measurement results.

1. はじめに

同一ドメインにおける組込みソフトウェア群の派生開発は、製品のシリーズ展開に伴いながら、高効率かつ一定の品質を保ったソフトウェアの開発を目的として実施される。しかしながら、派生元ソフトウェアの理解や変更が困難な場合、その後の派生開発において修正が多岐にわたる場

1) 早稲田大学グローバルソフトウェアエンジニアリング研究所
Global Software Engineering Laboratory, Waseda University
東京都新宿区大久保 3-4-1 Tel: 03-5286-3272 e-mail: washizaki@waseda.jp
3-4-1 Okubo, Shinjuku-ku, Tokyo, Japan
2) 株式会社小松製作所 開発本部・ICT 開発センタ
ICT Development Center, Development Division, Komatsu Ltd.
神奈川県平塚市四之宮 3-25-1
3-25-1 Shinomiya, Hiratsuka-shi, Kanagawa, Japan

合がある。それは結果として、開発効率や派生後のソフトウェアの品質を低下させ、派生の目的を十分に達成できない可能性がある。そこで派生を検討する元のソフトウェアについて、その後の派生開発にて修正を抑えた再利用がしやすいことを評価できることが望ましい。しかしながら、派生元ソフトウェアの静的な特徴が、その後の再利用にもたらす影響は十分には明らかとされていない。

そこで我々は、組込みソフトウェア群の派生開発を対象として、修正なしのブラックボックス的な再利用の可能性を予測する枠組みを提案する。本枠組みでは、修正なしの再利用に関係する C 言語プログラムソースコードの静的特徴を表すメトリクス群を用いて予測を行う。以降において、2 節で背景と研究課題を説明し、3 節で枠組みを提案する。4 節で枠組みを建機メーカの 10 の派生開発に適用した結果および得られた知見を説明し、5 節で関連研究について言及したのち、6 節で結論と展望を述べる。

2. 再利用性測定の必要性と研究課題

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メトリクスが提案されている[1][2][3][4][5][6][7]。それらの妥当性評価の多くは、一定規模のサンプルを特定観点（例えば再利用実績[1]や定性的評価[6-7]）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有効性を判定するものである。ただし、これらの方法では、対象プログラムの部分・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なデータに基づき評価を行う。従って、具体的かつ客観的な再利用実績と照らし合わせておらず、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。

また同様の理由により、個々のメトリクスが再利用性に与える影響の度合いを分析困難であった。複数のメトリクスを扱うにあたって、複数専門家の意見を集約するといった人手による定性的分析と重み付けの試みはある[8]。しかしながら、その実際の有効性は未検証であり、従って個々のメトリクスが再利用性に与える影響の度合いは定量的に明らかとされてこなかった。

さらに、過去の研究（例えば[9]）において調査対象として扱われるメトリクス候補は高々10 数個の少数のものに限られている。従って、ソースコードについて静的に測定可能な無数の特徴の中で特に重要なメトリクスを特定できていることの裏付けがなかった。

これらの問題の解決のため、本論文では以下の Research Question（研究課題）を設定する。ここで再利用性とは、ソースコードの以降の派生開発におけるブラックボックス的な修正なしの再利用のしやすさを指し、当該ソースコードに静的に固定された特性と捉える。従って再利用性が高いほど、当該ソースコードの以降における修正なしの再利用の実績が大きいという関係を期待するものとする。

- RQ1: ソースコードメトリクスにより以降の客観的な再利用実績を予測できるか？
- RQ2: 個々のメトリクスが再利用性に与える影響度合いを明らかとできるか？
- RQ3: 考えられる多数のメトリクス候補から裏付けを持って有用なものを特定できるか？

3. 再利用性測定の枠組み

我々は上述の研究課題を扱うため、再利用実績の予測を通じた再利用性の測定に有効と想定される複数のメトリクス（測定法）を定性的に識別する。そのうえで、測定値を具体的かつ客観的な再利用実績と照らし合わせることで、各メトリクスの有効性を精密に評価し RQ1 に応える。照らし合わせにあたり、実績を目的変数、各メトリクス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、各メトリクスが実績に与えている影響を精密に明らかとし RQ2 に応える。さらに、照らし合わせに用いるメトリクスとして、100 種類を超える多数のメトリクスを可能性のある候補として網羅的に挙げることで RQ3 に応える。

将来の派生開発における再利用のしやすさに影響する静的特徴を識別し、メトリクスとして選定し、測定に用いる枠組みを図 1 に示す。組込みソフトウェアでの派生開発を検討する際、この

枠組みにより導出された再利用性メトリクスを、派生元のソースコードに適用することで、再利用の程度や範囲、その再利用における修正必要性の予測が可能になることが期待される。

以降において、我々が提案する再利用性メトリクス群、および、再利用実績を述べる。

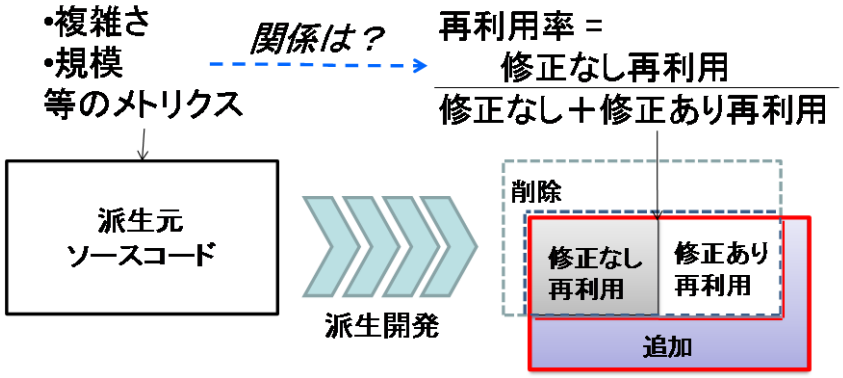

図 1: 派生開発と再利用性測定の枠組み

3.1 再利用性メトリクス候補の識別

我々は最初に、プログラムソースコードの静的解析に基づく品質診断ツール Adqua[10][11]が測定可能なメトリクスを中心として、ソフトウェアの理解や解析、変更、再利用に影響すると想定される 102 種類のソースコードメトリクスを候補として定性的に識別した。その抜粋を表 1 に示す。内訳は、関数を対象としたものが 3 種類（MFnXXX と表記）、ファイルを対象としたものが 91 種類（MFlXXX）、ディレクトリを対象としたものが 8 種類（MMdXXX）である。なお関数やファイルを対象としたメトリクスを、ディレクトリやシステム全体といった上位のモジュール単位の測定に用いたい場合は、個々の細かな測定結果を合計などにより集約する必要がある。

識別時点において、これらのメトリクスの測定値と実際の再利用のしやすさとの関係は未解明である。また、各メトリクスの生データをそのまま用いるとメトリクス間のデータの開きが大きくなってしまうため、適切な規模メトリクス（例えば関数に関するメトリクスであれば関数の数）の測定値で正規化した率データを用いることとした。

表 1: 検討した再利用性メトリクス候補（抜粋）

ID	名称	定義
MFl004	ステートメント数	言語の構文規則で、ステートメント(文)として認識されるものの数。 ステートメントとは、プログラム実行のための最小構成単位のこと。
MFl019	関数の数	ファイル内に定義されている関数の数。
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。
MFn042	関数内にてほかの関数を使用する回数	関数内にて他の関数を使用する回数。
MMd032	ディレクトリ外部から関数を呼ばれる自ディレクトリのファイルの率	ディレクトリ内の全ソースファイルのうち、ディレクトリ外部から関数を呼ばれるファイルが占める割合。
MMd042	依存するモジュールの数	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。

3.2 再利用実績の定義

派生元ソースコードの再利用の可否の判断には、当該ソースコードの分かりやすさや修正のしやすさよりも、新たな開発において課せられる要求や環境の都合が強く影響すると考えられる。

派生元ソースコードが再利用しやすいものであったとしても、製品のシリーズ展開にあたり要求や環境が大幅に変化すれば再利用し得ない。従って、再利用実績として全体のうちの派生元や派生先における再利用部分の割合を採用しても、派生元ソースコードの再利用のしやすさを適切に反映していない可能性がある。

そこで我々は、単純な再利用回数や割合・量ではなく、再利用された部分のうち修正なしで再利用された割合である再利用率[9]を、再利用の実績として用いる。再利用率が大きいほど、派生元のプロジェクト成果をブラックボックス的に再利用できたことを意味する。なお、再利用に対する労力や欠陥数などのプロセスデータを今回の対象について識別困難であったため、入手可能なプロジェクトの成果として、派生前後のソースコードを比較し分析することとした。

過去の研究[9]において、扱うメトリクス数は限られていたものの、C 言語プログラムソースコードの再利用が主に関数単位で行われていることが報告されている。そこで我々は、上述の再利用率について、関数を対象として次の式 1 により定義して用いることとした。

$$\text{再利用率} = \frac{\text{（修正なしに再利用された関数の数）}}{\text{（修正なし、もしくは修正を伴って再利用された関数の数）}} \qquad \text{（式 1）}$$

具体例を図 2 に示す。派生元のあるファイルに 5 つの関数 f1～f5 が存在する場合を考える。派生先の新規プロジェクトにおいて対応するファイル中では f1～f3 が再利用されている。そのうちで、修正なしでブラックボックス的に再利用されている関数が f3 の 1 つである。従って、当該ファイルにおける関数の再利用率は 1/3=33%となる。

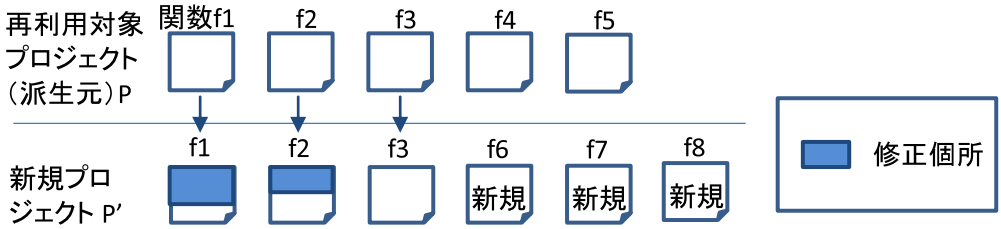

図 2: 再利用率の例

3.3 再利用性メトリクスの特定

続いて、一定規模の実際の組込みソフトウェアの派生開発における再利用率を再利用実績として測定する。そして再利用率を、派生元のプログラムソースコードに対する上述の全 102 種類のメトリクス候補の測定値と照らし合わせて、回帰分析により有効な再利用性メトリクスを統計的に特定する。

4. 枠組みの適用実験と知見

我々は上述の枠組みを実際の複数のプロジェクトデータに適用することで、再利用率の予測を通じた再利用性測定に有効なメトリクスを導出できるかどうかを検証した。以降において、測定値および再利用率を得る対象のプロジェクト、照らし合わせに基づく評価の過程、および、得られた結果を述べる。

4.1 測定対象プロジェクトの初期選定

我々は上述の枠組みを、建機メーカにおける実際の派生開発の実績データに適用した。具体的には、過去 1 度の派生開発が実施され、上述の再利用率を測定可能な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る 10 個のプロジェクト P1～P10 を選択し、それらから派生した新規プロジェクト P1’～P10’への再利用率を算出した。概要を表 2 に示す。派生開発後は派生前と比較して、関数の数に大きな差はないが、コード行数やファイル数が増大していることが分かる。

表 2: 測定・解析対象プロジェクトの基本情報

特徴	派生元	派生後
製品数	10	10
プログラミング言語	C	C、一部 C++
ソースコード行数	計 20 万 LOC 程度	計 30 万 LOC 程度
ファイル数	計 500 程度	計 1000 程度
関数の数	計 1 万程度	計 1 万程度

4.2 再利用実績の状況およびプロジェクトの絞り込み

3.2 節の定義に基づいて、関数を対象とした再利用率測定ツールを開発し、10 プロジェクトにおける再利用率を測定することで再利用実績を調査した。その結果を図 3 に示す。図 3 より、プロジェクトで再利用率が大きくばらついていること（最大 81%、最小 0%）、および、そもそも旧世代の全関数のうちで全く再利用がなされなかった派生開発プロジェクト（特に P10）が存在することが分かる。この結果は、製品のシリーズ展開における派生開発をもくろむ中で、その組込みソフトウェアをほぼ完全に作り直す場合があったことを示している。また、一口に派生開発といってもそもそも再利用するかどうかの方針がまちまちである。そこで、派生元のソースコードの特徴と照らし合わせるにあたっては、方針の違いの影響を強く受ける、全体のうちの再利用部分の割合ではなく、再利用率を用いることが適切と考えられる。

開発方法の追加調査の結果、P9, P10 については開発体制を刷新して異なる開発方法を導入しており、再利用率の低さは妥当なものと考えられる。また、P3, P5, P6 については、修正なしで再利用された関数の数が 1〜2 個と極端に少ないことが分かり、新たな派生開発における要求や環境の相違が大きいと考えられる。これらのデータを含めたままソースコードの特徴と再利用率の間の関係を分析すると、ソースコードの本来の再利用のしやすさを的確に捉えられない可能性がある。

そこで以降の分析においては、P1, P2, P4, P7 の 4 プロジェクトのデータを用いることとした。

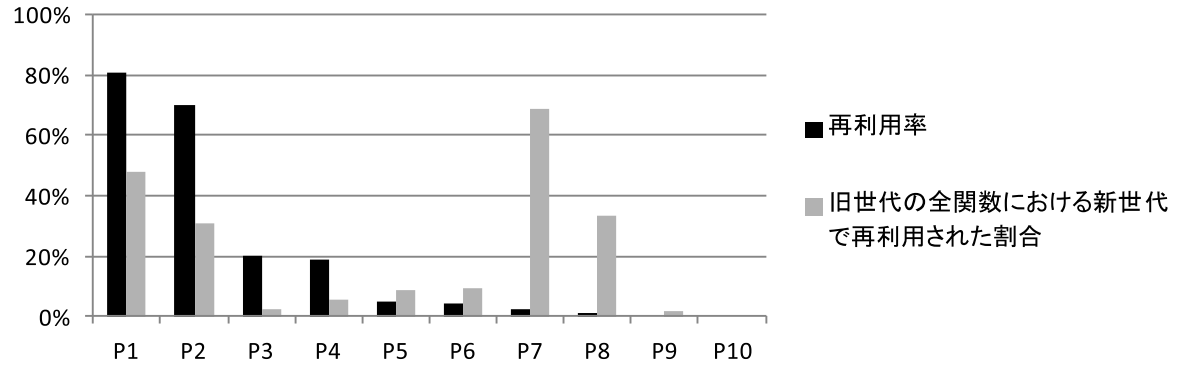

図 3: 全プロジェクトにおける関数の再利用率と再利用割合

4.3 分析による再利用性メトリクスの識別結果

続いて我々は、定性的に識別した 102 種類のソースコードメトリクスを P1, P2, P4, P7 に適用し、その率データと再利用率の関係を分析した。ファイル、ディレクトリ、システム全体の各モジュール単位において集約したそれぞれの再利用率を目的変数とし、各メトリクス群の率データを説明変数とした重回帰分析を繰り返し行い、自由度調整済み寄与率が最大となる組み合わせを特定した。多数のメトリクス群からの絞り込みに成功しており、提案する枠組みが RQ3 に応えられていることが分かる。

具体的には、ディレクトリ単位で以下の 5 つのメトリクスの率データからなる回帰式（式 2）を用いたとき、自由度調整済み寄与率が 0.82 と最大となり、重相関係数も 0.93 と大きいもので

あった。各メトリクスの係数はマイナスであるため、再利用率の向上のためには、各メトリクスの値を抑えればよいことが分かる。これらのメトリクスは主にモジュール間の結合に関係しており、主にヘッダファイルやグローバル変数を介した結合関係を弱くすればブラックボックス的に修正なしで再利用しやすくなることを示す。

再利用率=0.97 -1.63*MF1597-0.38*MF1588-0.28*MMd079-0.08*MF1190-0.01*MMd085　（式 2）

- 影響が相対的に強いもの
  - MF1597: 直接インクルードするヘッダファイルの種類数
  - MF1588: 他ファイルの外部結合グローバル変数を使用する関数の種類数
  - MMd079: 使用する外部結合グローバル変数が存在する外部ディレクトリの種類数
- 影響が相対的に弱いもの
  - MF1190: パラメータ数が 0 の関数の率
  - MMd085: ディレクトリ外部の外部結合グローバル変数を使用する自ディレクトリのファイルの種類数

例えば MF1588, MMd079 に着目すると、図 4 左側のように、多くの関数が他ファイルの外部結合グローバル変数を用いるようなファイルを含み、多くのディレクトリに依存するディレクトリ D1 の再利用性は低い。一方、図 4 右側のように、外部結合グローバル変数を用いる関数が絞られており、依存する外部ディレクトリ数も少ないディレクトリ D2 の再利用性は高い。

再利用率の実績値と、回帰式による予測値の散布図を図 5 に示す。図 5 から、一定の精度で再利用率を近似的に予測できており、提案する枠組みが RQ1 に応えられたことが分かる。また回帰式における係数により影響の強さを推測可能であり、RQ2 に応えられたことが分かる。

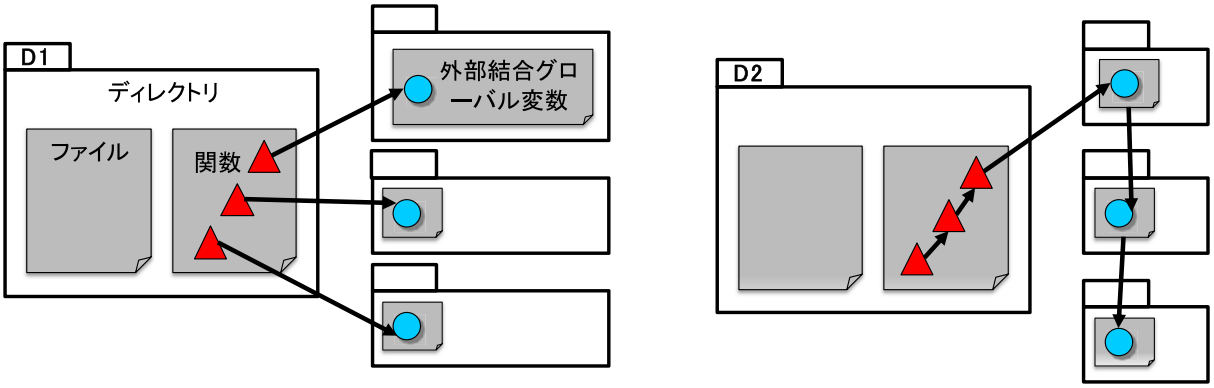

図 4: MF1588, MMd079 に基づいた再利用性の低いディレクトリ（左）と高いディレクトリ（右）

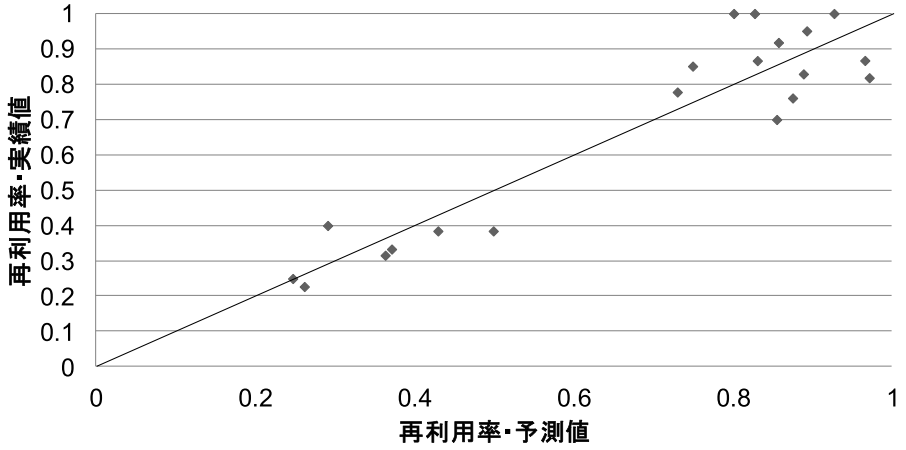

図 5: ディレクトリの再利用率の予測値と実績値（直線は予測値と実測値が一致するポイント）

4.4 再利用率予測の精度と得られた知見の活用

外部結合グローバル変数の使用に代表される、主にモジュール間の結合関係が、ブラックボックス的な修正なしの再利用の可否に影響していることを明らかとし、それらのメトリクスから再利用率を一定精度で予測する式を得た。この結果は、派生開発の検討における再利用率の予測や、それに基づく派生元の修正判断および以降の工数予測等に応用可能と考えられる。

また、将来において派生元となる可能性のあるソフトウェアの新規開発において、高い再利用性をあらかじめ作り込むための設計や実装の指針として参考になると考えられる。例えば、再利用性への影響の強いメトリクスを用いた指針を表 3 に示す。これらの指針を参照することで、ファイルやディレクトリの設計や実装において再利用性を作り込むことができる。

表 3: 再利用性を作り込むための設計・実装指針

メトリクス	対象単位	指針
MF1597	ファイル	直接インクルードするヘッダファイルの種類数を抑える。
MF1588	ファイル	外部結合グローバル変数の使用により多くの関数が他ファイルに依存しないように、依存する関数を少数にまとめる。
MMd079	ディレクトリ	外部結合グローバル変数の使用による外部ディレクトリへの依存関係を抑える。

4.5 妥当性評価

我々はプログラムソースコードの再利用性を反映する言わば「代理」として、派生前後における再利用率を用いた。我々は 3.2 節で定義した再利用率が派生元の再利用性を近似的に反映していると考えているが、要求の相違といった他の要因が再利用率に影響を与えている可能性はある。これは、内的な妥当性への脅威（Threats to Internal Validity）と捉える事ができる。そこで我々は今後、今回扱った 20 のプロジェクトの要求の類似性を調査する予定である。

外的な妥当性への脅威（Threats to External Validity）として、扱ったデータが属するドメインが限定的であることが挙げられる。本論文のプロジェクトデータはすべて建機組込みソフトウェアのドメインに属している。我々は、4.1 節におけるプロジェクトの基本情報などが、本論文において明らかとした知見および枠組みの有効範囲を一般化して、他のドメインにおける派生開発への適用可否を判断する上で参考になると考えている。また、本論文の主題は派生開発における再利用性を測定するものである。我々は今後、派生ではない新規の開発における過去のソースコード資産の再利用性の予測に対する知見の適合性について調査する予定である。

5. 関連研究

以前の研究[9]では、他の組込みドメインにおいて、再利用性評価に有効であると定性的に識別した 7 種のメトリクス候補を再利用実績に照らした分析により、以下の 2 種が有効な可能性があることを明らかとしている。具体的には、いずれのメトリクスについても値が小さい方が再利用実績が高いことを明らかとしている。

- 関数内にて他の関数を使用する回数
- ディレクトリ外部から関数を呼ばれるファイルが占める割合

本論文で識別したメトリクス群と上述の 2 種のメトリクスは、関数やファイルの結合関係に関係しているという意味で類似しており、ドメインが異なるものの類似した知見が得られたことが分かる。ただし本論文では、再利用性に影響を与える可能性のある 102 種のメトリクスを広く扱い、結果としてより多くの有効なメトリクスを特定することに成功している。

また[9]との相違の他の要因として、外部結合グローバル変数へのアクセス方法の相違が推測される。[9]の対象内で再利用実績の低いモジュールでは、モジュール外部の外部結合グローバル変

数へのアクセスにあたり、関数を介したアクセスの形を取るケースが多かったと推測される。一方、本論文の対象内で再利用実績の低いモジュールでは、直接的な変数アクセスの形を取る傾向にあったことが推測される。

なお、[9]や本論文の対象内で再利用実績の高いモジュールにおいてはそもそもの外部結合グローバル変数の利用が抑えられており、結果として上述の 2 種のメトリクス[9]や、4 節における 5 種のメトリクスが再利用性の測定に有効なものとして識別されたと考えられる。

6. おわりに

我々は、組込みソフトウェアの派生開発について、派生元のソースコードに対する種々のメトリクスの適用を通じて、以降の派生開発における再利用実績を予測する枠組みを提案した。扱った再利用実績は、派生元の再利用のしやすさを反映していると考えられる。建機メーカにおける 10 の実プロジェクトから選定したデータに枠組みを適用し、100 種類を超える多数のメトリクス候補から裏付けを持って再利用性測定に有用なものを特定することに成功した（RQ3）。さらに、個々のメトリクスが再利用性に与える影響度合いを明らかとしたうえで（RQ2）、以降の派生開発における再利用実績を予測することに成功した（RQ1）。

具体的には、外部結合グローバル変数の使用に代表される主にモジュール間結合関係が、ブラックボックス的な修正なしの再利用の可否に影響してくいることを明らかとし、それらのメトリクスから再利用率を一定精度で予測する式を得た。この結果は、派生開発の検討における再利用率の予測や、それに基づく派生元の修正判断および以降の工数予測等に応用可能と考えられる。

今後は、結果の一般性や有効な範囲を、[9]との相違の分析や、他企業・他ドメイン（組込み以外を含む）における検証を通じて明らかとしたい。

参考文献

[1] 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.
[2] G. Sindre et al., “The REBOOT Approach to Software Reuse,” Journal of Systems and Software, Vol.30, No.3, 1995.
[3] W. Frakes and C. Terry: “Software Reuse: Metrics and Models,” ACM Computing Surveys, Vol.28, No.2, 1996.
[4] L.H. Etzkorn et al.: “Automated Reusability Quality Analysis of OO Legacy Software,” Information and Software Technology, Vol.43, 2001.
[5] 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.
[6] H. Washizaki et al.: “A Metrics Suite for Measuring Reusability of Software Components,” 9th IEEE International Software Metrics Symposium, pp.211-223, 2003.
[7] 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.
[8] K. Lee and S.J. Lee: A Quantitative Software Quality Evaluation Model for the Artifacts of Component Based Development, Proc. 6th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, 2005.
[9] H. Washizaki et al.: ″Reusability Metrics for Program Source Code Written in C Language and Their Evaluation,″ 13th International Conference on Product-Focused Software Development and Process Improvement, LNCS Vol.7343, pp.89-103, 2012.
[10] H. Washizaki et al.: ″A Framework for Measuring and Evaluating Program Source Code Quality,″ 8th International Conference on Product-Focused Software Development and Process Improvement, LNCS Vol.4589, pp.284-299, 2007.
[11] 鷲崎弘宜, 田邉浩之, 小池利和: ″ソースコード解析による品質評価の仕組み″, 日経エレクトロニクス, 2010 年 1 月 25 日号, 2010.