# 品質定量指標のための異なるプログラミング言語間での規模の比較の試み

菊地 奈穂美† 平山 雅之† 出張 純也‡ 菊野 亨‡ 水野 修††
(独)情報処理推進機構† 大阪大学‡ 京都工芸繊維大学††
ソフトウェア・エンジニアリング・センター 大学院情報科学研究科 大学院工芸科学研究科

## 1. 目的と背景

ソフトウェア開発プロジェクトにおいて，作成するプロダクトやプロセスのデータを用いて指標値として表し，プロジェクトの進捗や品質のコントロールに活用することが有効である．プロダクトの指標では，開発する対象のソフトウェアの規模を使って正規化することが多い[1]．規模を使った指標で比較や判断していく場合に，正規化する際に母数として用いる規模でプログラミング言語を用いる際は言語種類を考慮する必要がある．

同じ仕様のソフトウェアの場合でも，プログラミング言語の違いによって規模の差が生じると，正規化後の指標の持つ意味が異なると考えるため，換算する等の考慮も必要となる．プログラミング言語別の規模は，FP と各言語の換算が公開されている[2][3]が，組込み分野でもエンタープライズ系でも多く使われているソースコード行数による比較情報の報告は殆ど無い．そのため，本研究では，プログラミング言語の違いによるソースコード規模の差を調査することとした．対象の言語としては，組込み，エンタープライズ両方でよく使われている主な言語から 3 種類の言語（C，C++，Java）での規模の差の特徴を調査した．調査に当たっては，異なる 3 種類の言語でプログラムを実際に開発する実験を行い，規模の差の評価を試みた．

## 2. 実験の概要

### 2.1 作成するプログラムの仕様

作成対象プログラム言語は C，C++，Java とした．

作成するプログラムの仕様は，公開情報から選定する方針のもとで，文献[5] 付録 D の演習問題を基に選定し，3 つの仕様としてまとめた(それぞれ Prg-1,Prg-2,Prg-3 とする）．プログラムの特性による偏りを少なくするため 3 つ選定した．各仕様に含めた内容を次に示す．

- Prg-1: リンクリストを用いて線形回帰パラメータを計算する．(文献[5] 付録 D のリンクリスト, 1A, 4A)
- Prg-2: ファイルへ数値の格納と検索をし，入力データチェック機能を持つ．(文献[5] 付録 D の 3B(1B, 2B の内容を含む)
- Prg-3: 複数プログラム行数と規模メトリクスを計算し出力する．(文献[5] 付録 D の 3A, 及び文献[1]ESQR の B10~B14 指標を出力)

エラー処理は共通仕様を決めた．もとにした文献[5]に記載なしであったため，異常な入力データでもプログラムがハングアップする等の異常終了せずにエラーを表示して終了することとした．

### 2.2 プログラム作成者と作成分担

被験者として 21～26 歳の 8 名がプログラムを作成した．被験者 7 名は 2 種類のプログラムをそれぞれ異なる言語で作成した．1 名のみ 1 種類の言語で 1 つ作成した．例えば，A さんは，Prg-1 を C 言語，Prg-2 を Java で作成した．各プログラム Prg-n は，1 人で 1 プログラムを作成した．各 Prg が各言語で 1～2 名で作成されるように被験者を割り当てた(表 1 を参照)．

被験者は，大学の講義相当によって基本的プログラミングスキルを習得済であり，各自が作成で用いた対象言語の基礎的知識は習得済であった．7 名は実用レベルの小規模プログラム作成経験を有していた．1 名のみ開発経験が 3 年程度とやや経験が多かった．

表 1. 各プログラムの言語と被験者

| プログラム | C | C++ | Java |
|---|---|---|---|
| Prg-1 | A,C | D | B,F |
| Prg-2 | B,D | G | A,E |
| Prg-3 | E,F | H | D,G |

### 2.3 開発環境・進め方

開発環境は，Redhat Linux OS 上，コンパイラは C と C++では gcc，Java では javac を用いた．

言語のライブラリは入出力用ライブラリを使用して良いこととし，開発時にプログラミング言語の書籍などを参照しても良いが，既にできているソースコードを書籍やインターネットなどから流用することは禁止した．複数名が同一仕様のプログラムを作成するが，実験条件として，被験者間で仕様の詳細内容の理解と確認のために仕様の議論はできるが，ソースコードはお互いに見ないことをルールとした．コードは毎日できたものを

Analysis of Source Code Size by Different Programming Language for the Use of Quantitative Quality Measures
† Nahomi Kikuchi , Masayuki Hirayama IPA/SEC
‡ Junya Debari, Tohru Kikuno, Osaka University
†† Osamu Mizuno, Kyoto Institute of Technology

CVS に登録する形で構成管理し，コードが出来た後のテストで検出した不具合をバグ管理票に記載した．完了条件として，標準的な入力データについてテストが通ることを事前に定義し，テストデータの例は提供した．実験は 2009 年 9 月～11 月に行なった．テスト完了後，作成したプログラムの概要を被験者にインタビューし，特徴を把握した．

## 3. 実験の結果

### 3.1 計測結果

文献[1]の指標のうち次のメトリクスを計測できた．*斜体の略号*は文献[1]の指標を示している．

・ソースコードの全行数 *TLOC*
・ファイル行数 *FLOC*
・Class と関数の行数　※文献[1]の *MLOC* は C 言語が前提で関数の行数のため，本実験は Class の行数と関数の行数の和とした．
・制御文数 *NOCS*
・コメント行数 *CLOC*
・関数/メソッド数 (※C++, Java のみ)
・クラス数 (※C++, Java のみ)
・ファイル数　物理的なファイルの数

3 つの仕様について 3 種類の言語で作成されたプログラムを計測したメトリクスの一部を表 2，表 3，表 4 に示す．

表 2. Prg-1 のメトリクス

| | Prg-11 | Prg-12 | Prg-13 | Prg-14 | Prg-15 |
|---|---|---|---|---|---|
| | C | C | C++ | Java | Java |
| ファイル数 | 1 | 1 | 1 | 3 | 4 |
| Classと 関数の行数 | 129 | 160 | 135 | 116 | 153 |
| 制御文数 | 23 | 18 | 21 | 16 | 21 |
| 関数/メソッド数 | 7 | 3 | 7 | 9 | 8 |
| クラス数 | - | - | 1 | 3 | 3 |

表 3. Prg-2 のメトリクス

| | Prg-21 | Prg-22 | Prg-23 | Prg-24 | Prg-25 |
|---|---|---|---|---|---|
| | C | C | C++ | Java | Java |
| ファイル数 | 1 | 1 | 1 | 2 | 1 |
| Classと 関数の行数 | 509 | 199 | 173 | 189 | 207 |
| 制御文数 | 146 | 65 | 61 | 49 | 52 |
| 関数/メソッド数 | 12 | 8 | 4 | 4 | 9 |
| クラス数 | - | - | 1 | 2 | 3 |

表 4. Prg-3 のメトリクス

| | Prg-31 | Prg-32 | Prg-33 | Prg-34 | Prg-35 |
|---|---|---|---|---|---|
| | C | C | C++ | Java | Java |
| ファイル数 | 1 | 1 | 20 | 1 | 4 |
| Classと 関数の行数 | 128 | 238 | 1464 | 251 | 484 |
| 制御文数 | 39 | 77 | 575 | 68 | 155 |
| 関数/メソッド数 | 2 | 3 | 38 | 9 | 13 |
| クラス数 | - | - | 11 | 1 | 4 |

### 3.2 考察

規模の差の背景を把握する為，作成者へのインタビュー及びコード内容の確認を行った所，次のことがわかった．

(1) C++の Prg-13,23, Java の Prg-14,15,24,25,34,は，オブジェクト指向(OO と略す)設計というよりは手続き型の関数呼出しでの C 言語のような実装に近い．一方，Prg-33,35 は OO 設計になっていた．Prg-35 は Java のライブラリを活用した構造であった．

(2) 同一仕様で同一言語で，規模の差が出ているものは，開発者による仕様の解釈の違いが出ていることをインタビューで確認できた．

(3) 表 4 の Prg-33 は OO 設計であり，拡張性が高い実装のため C++での規模も大きくなっていた．

(4) Java は，ライブラリ関数を活用した実装の場合(表 4 の Prg-35)，OO 設計であっても C++と比べ少ない行数であった．一方，Java でライブラリの活用をせずにコードで実現すると，C よりも行数が多くなっていた(表 3 の Prg-25)．

これらを確認したうえで，同じ仕様で作成されたプログラムの規模を比較してみると，同一仕様で異言語(C, C++, Java)で規模の差がある程度小さいものは，開発者による仕様解釈がほぼ同じで，かつオブジェクト指向言語でもオブジェクト指向設計ではない場合であった．次に，言語間の規模差を文献[2]と比較してみると，[2]では 1 FP と同程度のコード行数を公開しており最頻値では “1FP=C で 128 行，1FP=C++で 55 行 と 2 倍以上の差になっている．今回の Prg-1(表 2)では，C と C++は 2 倍の差は見られず，[2]とは異なる傾向になっている．

## 4. まとめ

本実験は比較的小規模の少数事例データであるが，言語間での規模の差の特徴と傾向について調査を試みた．プログラミング言語の違いだけで規模の差を説明することは難しく，ソフトウェア構造の違いなども規模の差として出てくることが調査によりわかった．

## 参考文献

[1] (独)情報処理推進機構 ソフトウェア・エンジニアリング・センター: 組込ソフトウェア開発向け品質作りこみガイド, 翔泳社(2008).
[2] Capers Jones: ソフトウェア開発の定量化手法 第 2 版, 共立出版(1998).
[3] http://www.psmsc.com/
[4] (独)情報処理推進機構 ソフトウェア・エンジニアリング・センター: ソフトウェア開発データ白書 2009, 日経 BP 社(2009).
[5] Watts Humphrey: パーソナルソフトウェアプロセス技法, 共立出版(1999).