THE IEICE TRANSACTIONS ON INFORMATION AND SYSTEMS (JAPANESE EDITION)

電子情報通信学会 論文誌 D 情報・システム

VOL. J99-D NO. 4
APRIL 2016

情報・システムソサイエティ

一般社団法人 電子情報通信学会
THE INFORMATION AND SYSTEMS SOCIETY
THE INSTITUTE OF ELECTRONICS, INFORMATION AND COMMUNICATION ENGINEERS

C と Java におけるライブラリ API の識別子名定義の頻度比較

神田　哲也[†a)]　　ダニエル　モラレス　ゲルマン[††,†]
石尾　　隆[†]（正員）　井上　克郎[†]（正員：フェロー）

Comparing Frequency of Identifier Definition in C and Java APIs

Tetsuya KANDA[†a)], Daniel M. GERMAN[††,†], *Nonmembers*, Takashi ISHIO[†], *Member*, and Katsuro INOUE[†], *Fellow*

[†] 大阪大学，吹田市
Osaka University, Suita-shi, 565–0871 Japan
[††] ヴィクトリア大学，カナダ
University of Victoria, Victoria, BC, Canada
a) E-mail: t-kanda@ist.osaka-u.ac.jp
DOI:10.14923/transinfj.2015JDL8026

あらまし　本論文では，C 言語と Java のライブラリを解析し，API の識別子名について定義頻度に傾向の違いがあることを明らかにした．

キーワード　API，識別子

1. まえがき

ソースコード中の識別子名はプログラム解析において重要な要素の一つである．例えばライブラリの API について，利用関係などを特定するために識別子名などを用いた解析が行われている．

Subramanian らは，技術フォーラム StackOverflow の投稿から抽出した Java と JavaScript のコード片に対して解析を行い，使用されているライブラリを特定した [1]．彼らはコード片に対して抽象構文木を構築し変数の型を特定した．Java においては，パッケージ名で名前空間を分割することができるため，同一のクラス名・メソッド名が複数存在する．この性質のため，コード片中の識別子がどの型であるかを特定するためには，字句ベースの解析だけではなく構文木ベースの解析が必要であるとしている．

一方，C 言語における名前空間は単一である．そのため，特に他のプログラムから呼び出されることを前提としたライブラリにおいては，識別子名が重複しないよう命名されていると予想できる．

本研究では，Java と C で書かれたライブラリを解析し，重複する識別子名の出現数を調査した．調査の結果，C で書かれたライブラリの識別子名は Java と比べ重複しない割合が高いことが判明した．

2. 調査手順

本研究では，C と Java で記述されたライブラリに含まれる識別子名を解析する．他のプログラムから呼び出されることを想定された識別子名が，ライブラリ間でどれほど重複しているかを測ることで，言語間での識別子命名の傾向の違いを明らかにする．

本調査では，C で書かれたライブラリを Linux ディストリビューションである Debian 7.5.0 のパッケージから収集した．また，Java で書かれたライブラリをライブラリ管理ツールである Maven のセントラルリポジトリから収集した．

識別子の一覧は，以下の手順に基づいて得る．

2.1 ライブラリファイルの収集

Debian のバイナリパッケージを解凍し，`/lib`，`/usr/lib`，`/usr/lib64` 以下に配置される拡張子が `.o`，`.so`，`.a` のバイナリファイルを取得する．Maven リポジトリからは，全ての jar ファイルを収集する．

2.2 識別子定義の抽出

Debian から抽出したバイナリファイルからは `readelf` コマンドを用いて，Maven に含まれる Jar ファイルからは，`javap` コマンドを用いて識別子定義を抽出する．この際，Jar ファイルから取り出した識別子からはパッケージ名を取り除く．Jar ファイルの中に含まれる Jar ファイルに関しては除外する．

2.3 無関係な識別子の除去

抽出した識別子定義から他言語で書かれたものやコンパイル時に埋め込まれたものを除去する．Debian の識別子定義から，対応するソースパッケージに含まれる C 言語で書かれたファイルに含まれない識別子定義を取り除く．Maven のライブラリについては，クラスファイルのコンパイル元ファイルの情報をもとに Java 以外で書かれたものを拡張子で判断し取り除く．

2.4 計　　測

C に関しては，ある識別子名がいくつのソースパッケージに定義されているかを数えた．Java に関しては，ある識別子名がいくつの Jar ファイルに定義されているかを数えた．Maven リポジトリには同一ライブラリの異なるバージョンが複数収録されているため，バージョン番号のみが違う Jar ファイルをまとめて一つのライブラリとして集計した．

3. 解析結果

表 1，表 2 に C 言語の関数名と変数名，表 3 から表 5 に Java のクラス名，メソッド名，フィールド名について調査結果をまとめた．表中「識別子数」は，識別子名の種類に対して数えた場合を，「識別子定義数」は全識別子定義に対して数えた場合を示している．例えば表 1 の「定義されているライブラリ数」が 2 の行は，全識別子名のうち 60,574 個が二つのライブラ

表 1　C 関 数 名
Table 1　C function names.

定義されている ライブラリ数	識別子数		識別子定義数	
1	886,134	91.80%	886,134	82.29%
2	60,574	6.28%	121,148	11.25%
3	13,483	1.40%	40,449	3.76%
4	2,571	0.27%	10,284	0.96%
5 以上	2,477	0.26%	18,824	1.75%
総数	965,239		1,076,839	

表 2　C 変 数 名
Table 2　C variable names.

定義されている ライブラリ数	識別子数		識別子定義数	
1	206,111	92.62%	206,111	82.14%
2	10,646	4.78%	21,292	8.49%
3	3,974	1.79%	11,922	4.75%
4	926	0.42%	3,704	1.48%
5 以上	875	0.39%	7,904	3.14%
計	222,532		250,933	

表 3　Java クラス名
Table 3　Java class names.

定義されている ライブラリ数	識別子数		識別子定義数	
1	445,353	75.85%	564,463	43.44%
2	87,538	14.75%	228,600	17.59%
3	27,217	4.59%	107,338	8.26%
4	11,240	1.89%	59,901	4.61%
5 以上	22,067	3.72%	338,991	26.09%
計	593,415		1,299,293	

表 4　Java メソッド名
Table 4　Java method names.

定義されている ライブラリ数	識別子数		識別子定義数	
1	1,010,135	66.03%	2,240,606	15.75%
2	260,930	17.06%	1,269,538	8.93%
3	92,622	6.05%	978,365	6.88%
4	48,160	3.15%	536,944	3.78%
5 以上	117,860	7.70%	9,196,312	64.66%
計	1,529,707		14,221,765	

表 5　Java フィールド名
Table 5　Java field names.

定義されている ライブラリ数	識別子数		識別子定義数	
1	387,149	70.28%	565,455	29.21%
2	83,367	15.13%	272,301	14.07%
3	33,335	6.05%	179,962	9.30%
4	16,059	2.92%	118,384	6.11%
5 以上	30,951	5.62%	799,915	41.32%
計	550,861		1,936,017	

リにおいて定義されており，識別子定義の数としては 121,148 個がここに分類される．

一つのライブラリにのみ定義されている識別子名は，C の関数名と変数名で約 9 割，Java のクラス名とメソッド名，フィールド名で約 7 割であった．C と Java 双方で特定のライブラリにしか存在しない識別子名が多く，複数のライブラリに重複して宣言される識別子名は少ないことがこの結果から読み取れる．

一方，全識別子定義に対しての割合を数えると，言語間で大きな差が見られた．C では 8 割の識別子定義が一つのライブラリにのみ定義されていたのに対し，Java ではその割合は低く，特にメソッド名では 2 割を切った．このことから，Java のメソッド名では一部の識別子名が多くのライブラリにおいて宣言されているという偏りがあることがわかった．

複数のライブラリに定義される識別子を調べたところ，ツールによるソースコードの自動生成，外部のライブラリの取り込み，共通のフレームワークの利用，といった理由によるものが見られた．C 言語のライブラリから多く見つかった文字列 “yy” から始まる識別子は，Yacc パーサジェネレータによって自動的に生成されたソースコードに含まれる関数群である．また，`strlcpy` や `strlcat` といった関数は Linux に含まれない BSD libc において定義されている関数を幾つかのライブラリが独自に取り込んだと予想される．共通のフレームワークを拡張して利用している場合，例えば Java において多く定義されていた `Activator` クラスは，ほとんどが `org.osgi.framework.BundleActiovator` インタフェースを実装したものであった．このインタフェースには必須のメソッド `start` や `stop` を実装する必要があるため，これらのメソッド名も複数のライブラリで定義されていた．

4. 応用への展望

ソースコードから識別子定義のみを抽出することは，型や依存関係を含めたソースコード解析に比べ非常に軽量な処理である．特定のライブラリにのみ宣言さる識別子が多かったことから，偏りの少ない C 言語においては識別子名のみでソースコードから利用しているライブラリを特定することができると期待できる．また，重複して宣言されていた識別子名も含め，どのような識別子名が定義されているかを列挙することで，ライブラリのコピーを検出することなどが可能になると考えられる．

謝辞　本研究は科研費（25220003）及び大阪大学国際共同研究促進プログラムの助成を得た．

文　献

[1] S. Subramanian, L. Inozemtseva, and R. Holmes, “Live API documentation,” Proc. ICSE2014, pp.643–652, 2014.

（平成 27 年 10 月 15 日受付，12 月 2 日再受付，12 月 29 日早期公開）