ソフトウエア工学 93−19
（1993．7．9）

クラスライブラリの理解容易性の評価について

中西弘毅 荒野高志 三宅武司
NTT ソフトウェア研究所
nakanisi@nttslb.ntt.jp

概要:
本稿では、自然言語の研究成果を応用したクラスライブラリの理解容易性評価方法の提案を行なう。まずZipfの法則という自然言語の研究過程で発見された法則が幾つかの定評のあるオブジェクト指向クラスライブラリに対して成立することを示す。次に、Zipfの法則に含まれるパラメータとクラス階層との間に相関関係があることを統計的に示し、クラス階層とZipfの法則の関係について考察する。最後に、これらを利用して、クラスライブラリの理解容易性評価尺度の提案を行なう。なお今回分析対象としたのは、C++で作られた実績のあるクラスライブラリInterViews,ET++である。

Evaluating the Understandability of Class Libraries

Koki Nakanishi, Takashi Arano, and Takeshi Miyake
NTT Software Laboratories

Abstract:

This paper proposes a way of evaluating the understandability of class libraries.At first, Zipf's law, an established law in linguistics that measures the effort needed to read text, is verified to some well respected object-oriented class libraries. In the next place, it is shown that class hierarchy affects a parameter of Zipf's law. By using this phenomena, a measure of understandability of class libraries is derived. In this experiment, InterViews and ET++ which are well respected class libraries are used.

1 はじめに

オブジェクト指向法を適用することによって、保守性、信頼性、生産性の高い開発が可能であると考えられている [1]。これは、特にオブジェクト指向法の持つ継承機能によるところが大きい。

継承機能は、is-a関係、kind-of関係を階層構造として入れ、個々の構成要素の増加に伴う複雑さを抑制する働きを持つ。これによる一つの効用として、クラスライブラリの理解容易性が挙げられる。本研究の目的は、継承機能に起因したクラスライブラリの理解容易性評価尺度を提案することである。

そこで、理解容易性に対して継承が有効に用いられているか否かを計るのであるが、理解容易性というものが非常に概念的なものであるために、これを実現する上で次のような二つの問題がある。一つは、継承が幾つ用いられたかを計ること [2] が必ずしも理解容易性を示すことにはならないこと。もう一つは、開発環境の持つ多様性と人的要因に起因する問題である。理解に要した時間、理解テストの結果などによって理解容易性を計った場合、これらの数にはソフトウェアの構造が直接反映されていない。そこで、継承を定量化した数値とこれらの値を比較して有効性を導くことになる。例えば、インヘリタンスグラフの深さと、理解度テストをして実験した結果を比較したものがある [3]。しかし、この様な方法は、開発環境、個人差などの要因に依存していないことを示すことが難しく、再現性に問題が起こる。

本研究では、クラスライブラリの記述に用いられた言葉の出現頻度を利用する。言葉の出現頻度を採ることによって、一般のテキストと同等に扱うことが可能となる。これによって、自然言語、心理学の成果を適用した理解容易性の議論が可能となる。この様にして得られた結果は、開発環境に依存せず、そしてまた継承を直接計ったものでもない。よって先に挙げた二つの問題を解決している。更に、適用に関してはフィードバックを掛けることが容易であるという利点がある。

第一章では、幾つかの定評のあるクラスライブラリに対して出現頻度の分布を計測し、それが自然言語において発見されたジップの法則を満たしていることを示す。第二章では、継承を用いることが、このジップの法則の成立性、及びそれに含まれるパラメータに対して、大きな影響を与えていることを実際に開発されたクラスライブラリを計測対象として示す。最後の章で、再利用性、保守性に対して大きな影響を与えると考えられるソフトウェアの理解容易性と言葉の出現頻度の分布がどのような関係にあるかについて考察する。

2 クラスライブラリに対するジップの法則の検証

2.1 ジップの法則

この章では、テキストに現われる言葉の出現頻度とその順位の間の関係を表すZipfの法則 [4][6][7] について述べる。Zipfの法則とは、テキストで使われている言葉を出現回数の多い順に並べると出現回数がほぼその順位に反比例するというものであり、式にして表すと次のようになる。

$$P_r \equiv \frac{n_r}{n} = \frac{0.1}{r}$$

ここで、rは出現順位、P_r はr番目の単語の出現確率 、n_r は出現回数、nは全単語の総出現回数である。なを、この式は近似式であり、一定の数を越えると総和が1を越えてしまう。Zipfの法則は始め、Estoup,Condonによって最初に発見され、Zipfがそれを最小労力と関連させて紹介し広く知られるようになった。その後、この法則をB.Mandelbrotは更に正確なものとするため、次のように一般化した。

$$P_r \equiv \frac{n_r}{n} = \frac{C}{r^B}$$

B、Cは定数である。この時、BとCは出現確率の総和が1になるように決定する。本研究ではこの一般形の方を用い以後これをZipfの法則ということにする。この式は、両辺の対数をとると

$$\log P_r = \log C - B \log r$$

となり両対数座標でグラフに表すと直線になる。それを、図に表したものが図1である。ここでBの意味は、自然言語の分野で研究されそれについては第4章でクラスライブラリの理解容易性との関連で詳しく述べる。

図2はジェームス・ジョイスの「ユリシーズ」の260,430個の連続した単語（曲線A）と、新聞からの43,989個の連続した単語（曲線B）について、出現頻度（各単語が使われた回数）を順位（出現頻度の大きなものからの順番）に対してプロットしたものであり、直線CはZipfの法則の理想曲線である [4]。

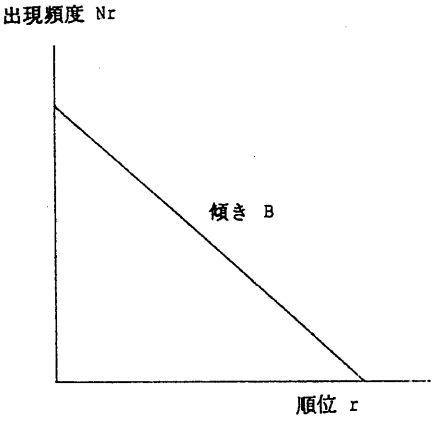

図 1: Zipf の法則の理想曲線

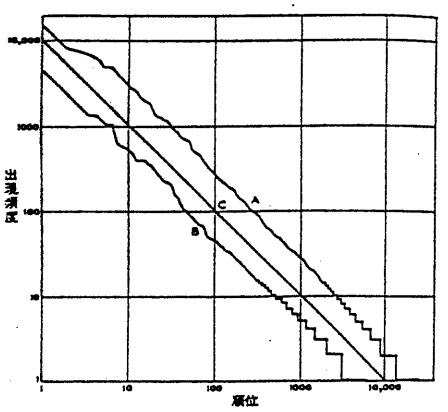

図 2: 自然言語における出現頻度とその順位の分布 (byZipf)

自然言語以外の分野では、ジップの法則について次のような研究が行なわれている。

- PL/1 について単語の出現回数を計測した例。 [7]
- EMACS というエディタのコマンドの利用頻度 [9][10]
- 「後藤・野島１９９２」 [8] では、知識の経済学を展開しそこで用いられる効用関数として Zipf の曲線が採用されている。

2.2 クラスライブラリに対する Zipf の法則の検証

今回計測対象としたクラスライブラリは InterViews、ET++ である。これらは、C++ で記述されたウィンドウアプリケーション開発用クラスライブラリで、クラスライブラリとして定評のあるものである。

C++ では、クラスライブラリのソースは.h ファイルと.c ファイルに分けられている。.h ファイルではクラスライブラリに含まれるクラス名と継承関係、及びそのクラスのメソッド（メンバー関数）名とその簡単な性質（void など）が記述されている。一方、.c ファイルには.h ファイルにあるメソッドの実装が記述されている。.h ファイルは更に public、protected、private の三つの部分に分けることが出来る。public 部にはどのクラスからも利用できるメソッドが記述され、protected 部は自分自身のクラスの中から及び直接継承関係にある子クラスから利用可能なメソッド、private 部はクラスの中からのみ利用できるメソッドが記述されている。つまり、情報隠蔽という観点から C++ では、他のクラスは public 部に書いてあるメンバー関数にしかアクセスすることは出来ない。

よって、利用者から見たクラスライブラリの複雑さは、public 部に反映される。そそこで、.h ファイルの public 部を計測対象とする。また、ポインタ"*", アドレス"&"、"{"、"}"、":"、";" 及び消滅子はそれから得られる情報はクラスライブラリの利用者から見た時、その概要を理解する上で重要ではないと考え計測対象から除いた。計測した結果を、図 3 に示す。

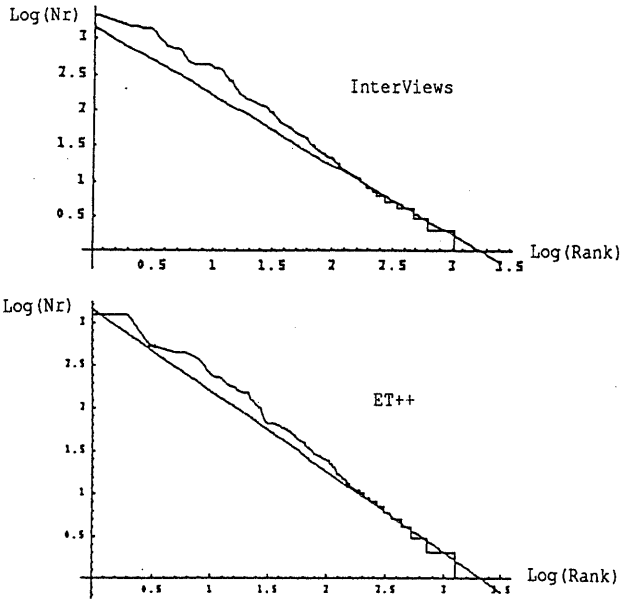

図 3: クラスライブラリにおける出現頻度 vs 順位（頻度における）

図から Zipf の法則が良く成り立っていることが判る。

出現頻度と出現頻度の大きさに関する順位を対数変換すると、Zipf の法則によればこの二つの量は比例することになるが、二つの値の実際の相関係数は InterViews が 0.974、ET++ が 0.975 となり、Zipf の法則がかなり良く成立していることをが判る。

3 ジップの法則と継承

3.1 Zipfの法則と継承との相関関係

この章では、先のInterViewsクラスライブラリを機能別に幾つかのグループに分類し、各グループの継承による階層化の度合とZipfの曲線の傾きを表すパラメータの関係を分析する。InteViewsクラスライブラリは、次のように6個のグループに分けられている。

- InterViews — Intrinsics
- IV–2_6 — Class for compatibility with 2.6
- IV–X11 — X11–dependent implementation classes
- IV–look — concrete user interface classes
- Unidraw/Graphics — drawing framework
- OS— operating system support classes

上の6個のグループの中でIV-X11のグループは他のグループと比較した時、継承がほとんど用いられていないという特徴を持っている。次にこのことが、Zipfの法則とどのような関係にあるかを見る。個々の6個のグループに対する出現頻度と出現頻度を大きい順に並べた時の順位を両対数グラフに表したのが図4である。

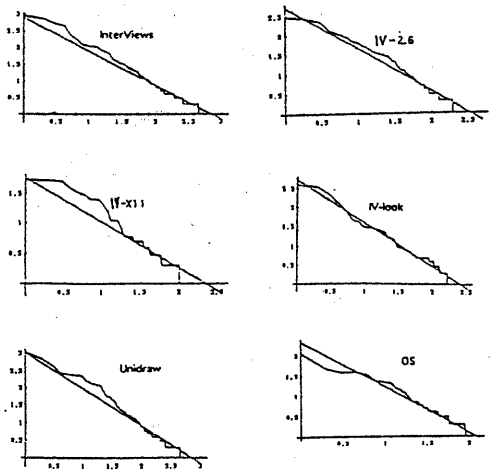

図4: 各グループの出現順位vs順位

この図を見て判るように、Zipfの法則から求まる直線からIV-X11は他のグループと比較して大きく外れていることが判る。更に、これらのグループに対してZipfの法則を対数グラフに表した時の傾きと語彙数の関係を表したのが図5である。ただし、横軸に規模を示したのは、図から判るように語彙数が傾きに影響を与えることを考慮したからである。この図を見ると、IV-X11とその他の5個のグループでは傾きの絶対値の値に大きな差があることが判る。つまり、継承を用いることがZipfの法則のパラメータBの大きさに大きな影響を与えていることが予測される。

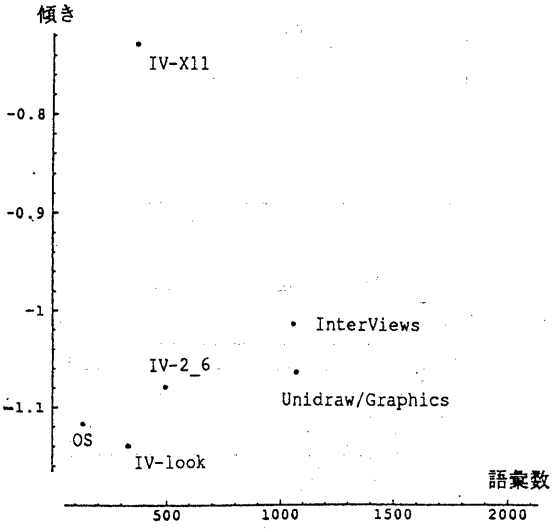

図5: Zipf曲線の傾き vs 語彙数

3.2 ジップの法則と継承との関連

オブジェクト指向技術では汎用性の高い抽象クラスを作ることがクラスライブラリの再利用性を高めると考えられている[1]。その際重要なことは、適宜それまでに作った既存のクラスを見直し、それらに、共通した性質、機能を抽出することが重要となる[11]。これは、普段誰もが使っている一般化と特殊化の作業である。それによって、クラスライブラリは一貫性を持ち、再利用する際に重要となる知識の理解が容易になる。この時、上のような作業によって具体的に目に見える作業は次のようなものである。似たような属性が、クラスによって異なる名前で定義されている。この場合同じ名前を割り当てて、それを共通の祖先クラスに移動する。これによって、その属性を参照する操作は、一貫性を持つことが出来る。同様に、似たような操作で違う名前を持ったものがないかを調べる。類似性を見失わないためには、名前付けの方針が一貫していることが重要である[11]。例えば、円、三角形、直線などに対して図形というスーパークラスを設けて、共通した振舞いである拡大、移動、回転という操作を図形クラスに定義にすることにより、個々のサブクラ

スに対するこれらの操作は類推によって理解することが可能となり、個別に理解する必要がなくなる。この際、拡大、回転、移動というメンバー関数はこれらの個々のクラスの.hファイルのpublic部で公開される。この様に、オブジェクト指向技術において名前付けの作業は最も重要な作業の一つとなる。オブジェクト指向の利点を生かしたより良いクラスライブラリを作るために行なわれた作業は名前付けに大きく反映される。これは、当然名前の頻度の分布にも大きな影響を与える。つまり、名前が整理され類似した意味を持つものは一つに統合される。これが、継承を用いることが、Zipfの法則を対数グラフに表した直線の傾きの絶対値を大きくしている原因である。

4 Zipfの法則と理解容易性についての考察

Miller[5]の研究によれば子供が成長するに連れてZipfの法則に現れる直線の傾きの絶対値は1.6から1.16に減少し、新聞、ジェームス・ジョイスの作品ではほぼ1になることが知られている。子供の場合に傾きが大きくなるのは、少ない限られた語彙を、多様な意味で用いることによって補っていることと関係していることが判っている。一方、クラスライブラリの利用者から見ると専門的な知識を蓄えるに必要となる労力を最小限に抑へた上でより高機能なサービスが受けられることが望ましい。継承を用いれば、メッセージはソフトウェアの側で、自動的にその文脈にあった関数によって実行される。これにより高度なサービスを利用者は意識することなく受けることが出来る。Zipfの直線の傾きの絶対値が大きくなることは、同じ名前で実現が異なるメソッドが多く存在していることを示す。これは、利用者が同じ名前の関数、クラスに対してその文脈にあった機能がクラスライブラリの側で選択されていることの現われである。よって傾きの絶対値の大きさは、クラスライブラリの利用者から見たわかり易さの指標として採用することができると考えている。つまり、「はじめに」で述べたように自然言語の成果を利用した特殊な環境に依存しないクラスライブラリの理解容易性の評価が可能となる。

5 謝辞

本研究を進めるに当たって適切な指導、助言をいただいた北陸先端科学技術大学院大学落水浩一郎教授、NTTソフトウェア研究所今瀬 真主幹研究員、古山 恒夫主幹研究員、西山 茂主幹研究員、後藤 滋樹ソフトウェア基礎技術部部長、クラスライブラリの利用方法について助言して下さったソフトウェア研究所のメンバに感謝します。

参考文献

[1] Bertrand. Meyer,"Object-oriented Software Construction". Prentice-Hall,1988.(邦訳:「オブジェクト指向入門」二木厚吉 監訳/酒匂 寛・酒匂 順子 共訳 アスキー出版局 1990)

[2] Shyam R.Chidamber."Towards a Metrics suite for Object Oriented Design" OOPSLA'91,pp.197-211,1991

[3] Al Lake,Curtis Cook."A Software Complexity Metric for C++"ANNUAL OREGON WORKSHOP ON SOFTWARE METRICS,1992

[4] George K.Zipf."Human Behavior and the Principle of Least Effort:An Introduction to Human Ecology" Addison-Wesley,Reading,Mass.1949

[5] George A.Miller ."Language and Communication" MaGraw-Hill,New York,1951.

[6] Jhon R.Pierce ."An Introduction to Information Theory:Symbols,Signals and Noise".Dover,1980.(鎮目恭夫 訳「信号・シグナル・ノイズ:情報理論入門」白揚社 1988年)

[7] Martin L. Showman ."Softoware Engineering".MaGraw-Hill pp.154-188 (箱崎勝也 訳「ソフトウェアの性能とテスト」マグロウヒル 1989年)

[8] 後藤滋樹 野島久雄 "情報流通機構の均衡分析" 人口知能学会学会誌 vol.8 No.5 May pp348-356 1993

[9] 奥乃 博 "画面エディタ Emacs のユーザー特性について". 情報処理学会第25回 プログラミングシンポジウム 164-173,1984年1月

[10] 奥乃 博."オンライン・プログラミングにおける問題解決へのリソースの分配" 日本認知科学会第1回大会発表論文集 A-7,1984年6月

[11] J.Runmbaugh,M.Blaha,W.Premerlani,F.Eddy, and W.Lorensen."Object-Oriented Modeling and Design" Prentice Hall,1991 (邦訳:「オブジェクト指向方法論 OMT」羽生田 栄一 監訳/トッパン 1992)