クラスライブラリにおけるZipfの法則の検証

5J－10

中西 弘毅　荒野 高志

NTTソフトウェア研究所

概要

本稿では、自然言語の研究成果であるジップの法則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。オブジェクト指向方法論では継承機構によって、類似性に基付いた体系的なソフトウェア部品の構築が可能となり、これまで以上のソフトウェアの部品化が可能となると考えられている。本研究では、ソフトウェア部品の利用者が部品を理解する上で必要となる労力を知識の効用関数として定義し、クラスライブラリにおける知識の効用関数がジップの法則を満たしていること実験的に明らかにする。このことを利用して先に述べた継承機能によって得られる効果をジップの法則に現れるパラメータによって評価する方法を提案し、検証する。

1　はじめに

オブジェクト指向方法論によって、ソフトウェア工学の一つの課題であるソフトウェアの部品化が現実しつつある。事実、InterViewsやET++などのソフトウェア部品の集合であるクラスライブラリもかなり多く見られるようになった。しかし、そこでは新たにライブラリの規模の問題が起こっている。つまり、提供されるソフトウエア部品の数が増大することによって、部品を探すことが困難になるという問題、そしてそれらを使いこなすために多くの労力を費やしてクラスライブラリを理解しなければならないという問題である。

一般に、このような膨大な量の情報を効率良く理解するための解決策は、各要素の類似性に着目し、それらを体系立てて整理することである。オブジェクト指向方法論では、これを実現するために継承機能が大きな役割を果たす。つまり、ソフトウェア部品の集合であるクラスライブラリの設計者はクラスによって違った名前を持ち同じ役割を果たしている関数を探し出しそれらの共通した関数（インターフェイスまたは振舞い）を持った抽象的なクラスを新しく設けてクラスを階層化することによって部品を体系立て、整理し、それまで個別に費やされていた理解に要する労力を節約することが可能となる。

本研究は部品の理解に費やされる労力を定量化しそれを用いた継承機能の評価メトリクスを提案し検証する。そのための手段としてジップの法則という自然言語の研究過程で発見された法則を利用する。

2　知識の効用関数

理解に要する労力を知識の効用関数として定義する。まず、効用関数とは、財(goods)を1単位だけ増加させた時の効用(utility)の増加分の意味である。図に表すと、横軸に一単位移動することが財を1単位増加させたことを表し、その時の縦軸の値が効用となる。一般に限界効用関数とは図1のような右下がりの曲線になることが、限界効用の逓減の法則として知られている。つまり、一般に財はもっとも効用の高いものから優先的に使われていくことを意味している。但し、右下がりの曲線であることは判っているが更にそれ以上のことをは、知識の種類によって異なる。後藤・野島[1]は経済学の基本的な原理である「需要と供給の均衡の法則」を知識の経済学として応用している。上の知識の効用関数の考え方は後藤・野島によるものである。

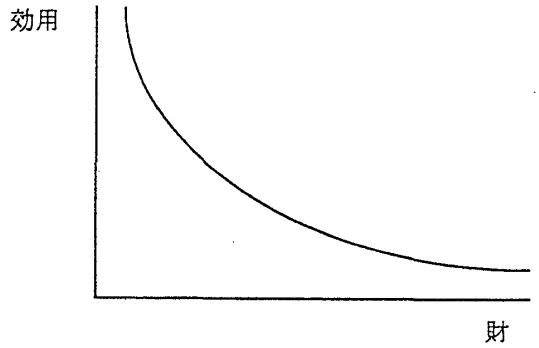

図 1: 一般的効用曲線

次に、利用者から見たクラスライブラリにおける知識の効用関数を定義する。オブジェクト指向法論では互いに類似したクラスClassA、ClassBが実装は異なるが意味的には同じメソッドを持っていたならばそれに対して同じ名前(Method1)を付けることができる。この時、このメソッドの名前Method1は二つのクラスClassA,ClassBに対して有効である。つまりクラスの型に応じて適切な実装が自動的に選ばれる。よってクラスの利用者はこのメソッドMethod1を覚えたことによって2の効用を得たと考えることができる。この様にして、クラスライブラリにおける効用関数は、縦軸つまり効用がそのメソッドが実装されているクラスの数、横軸がその数における順位となる。

そこで、クラスライブラリにおける知識の効用関数がどのような性質を持つかを分析する。前にも書いたように効用関数は右下がりの曲線になることは自明であるが、それ以上の性質は適用対象に依存する。

InterViewsというGUIアプリケーション開発用のクラスライブラリに対して効用関数を計測し、それを両対数座標グラフ上に表したものが図2である。見て判るように、効用曲線が直線状に並んでいる。これ以外にもg++のクラスライブラリ、nihcl-3.0、ET++に対しても同様に効用関数を両対数グラフ上に表したものが直線上に並に並ぶという性質を持っていることを確認した。これは、自然言語で知られているジップの法則というものである。それは、メソッド名にそのメソッドが実装されているクラスの回数に従って順位rを付け、r番目のメソッド名に対しそれが実装されているクラスの数をN_rとすると、ジップの法則は次の式が近似的に成り立つというものである。

$$N_r \cong \frac{C}{r^B}. \qquad (1)$$

Validation of Zipf's law for class libraries

Koki Nakanishi　Takashi Arano

NTT Software Laboratories

ここで、BとCは、計測対象に依存した定数である。両辺を対数変換すると

$$\log N_r \cong \log C - B \log r. \qquad (2)$$

となり、両対数座標軸上で直線となる。つまり、図2はクラスライブラリにおける知識の効用関数がジップの法則を満たしていることを表している。

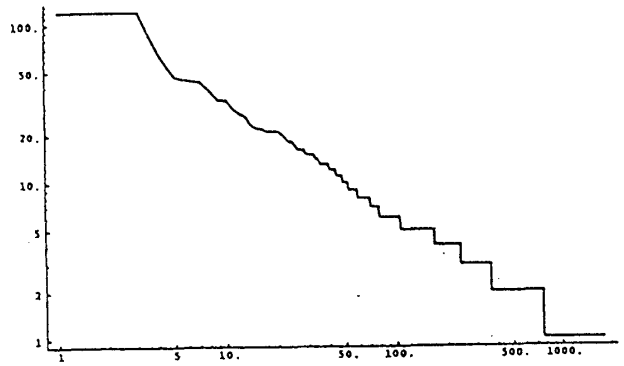

図 2: クラスライブラリの対数座標上の知識の効用曲線

なを、自然言語におけるジップの法則とは、文章における単語の出現回数とそれに基づく順位の関係が近似的に上の方程式で表されることをいう。またEmacsエディタのコマンドとその使用頻度も近似的に上の方程式で表されることが判っている。

3 継承とジップの法則

定義から、効用関数のグラフが左によるほど少ない財で多くの効果が得られることを意味することが解る。つまり、効用関数が左によるということは、クラスライブラリの利用者は覚えるメソッドが少ない数で多くの実装を使いこなすことができるということである。一方、設計者は、類似したクラスがないか、違った名前で同じ意味を持ったメソッドがないか、確認する[6]。もしそのようなことがあった場合、それを継承機能によって同じ名前割り当て共通の先祖クラスに設ける。継承機能によってクラスとメソッドの定義をうまく調節することがメソッドの名前を洗練させる。これによって効用関数のグラフを左に移動させる。よって効用関数によって継承機能の評価をすることが可能となる。今、クラスライブラリの知識の効用関数はジップの法則を満たしていることが判っているので、グラフが左によるということは傾きが大きくなることを意味している。つまり、効用関数の傾きを用いて継承機能の評価が可能となる。

4 検証

この章では、実際のクラスライブラリに対して傾きを評価尺度として適用した結果を述べる。InterViewsクラスライブラリは次の6個のグループから構成されている。

- InterViews – intrinsic kernel.
- IV-2.6 – classes for maintaining version compatibility.
- IV-X11 – X11-dependent implementation classes.
- IV-look – concrete user interface classes.
- Unidraw/graphic – drawing framework.
- OS – operating system support classes.

これらのグループの知識の効用関数が、ジップの法則を満たしていることを確認した。そこで、個々のグループのジップの直線（知識の効用関数）の傾きを示したものが図3である。グラフを見て解るようにIV-X11は他のグループと比較して、傾きが非常に小さい。一方、IV-X11は他のグループと比較して継承がほとんど使われていないことが判っている。つまりこのデータは継承を使うことによって傾きが大きくなる実例となっている。また、Unidrawは図形を扱うものでこれらのグループの中で最も完成度が高いと考えられる。つまり、図形を扱うクラスは、一次元図形（直線、折れ線、曲線）、二次元図形（多角形、円）、回転、移動、拡大、長さ、面積のように数学的に抽象的に定義され、成熟した分類体系を持っている。一方、Unidrawの傾きもこれらの中で最も高い。これら二つの例は、継承機能によって各クラスの概念が整理され、それによって知識の効用関数が変化し、それがクラスライブラリの場合ジップの法則に現れる直線の傾きの変化として現れたことを経験的に示している。

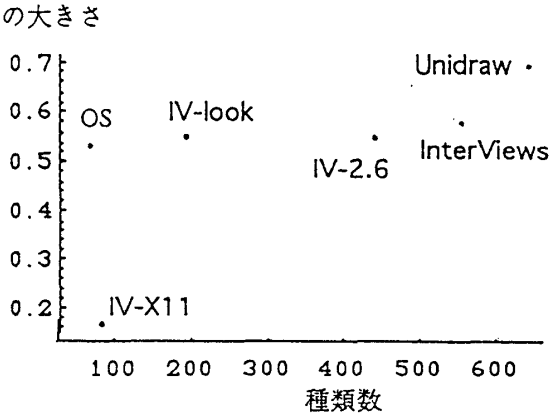

図 3: ジップの直線の傾きの分布

5 考察

これまでにも、継承を計るまたは評価するメトリクスは幾つか提案されている。それらの中で代表的なものは、継承グラフの深さや広さを基にしたものである[3]。しかし、これらのメトリクスはクラスのインターフェイスまでは考慮されていない。つまり、継承が各クラス、関数の類似性に基づいて行なわれたものか否かは判断できない。本来は、クラス階層を構成することによって、クラスと操作がどの程度整理されたかを計るべきである。ジップの直線における傾きはクラス階層を計っているのではなくクラス階層を作ったことによってクラスのインターフェイスが旨く整理され統制がとれているか否かを計るものとして、クラス階層の深さや広さに比べてより適している。

参考文献

[1] 後藤滋樹 野島久雄 “人間社会の情報流通における三段構造の分析” 人工知能学会学会誌 vol.8 No.3 May pp348-356 1993

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

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

[4] 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 pp.164-173, 1984年1月

[5] 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

[6] J. Rumbaugh, 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)