ソフトウェア部品グラフの次数分布におけるべき乗則の調査

市井　誠[†]　松下　誠[†]　井上 克郎[†]

[†] 大阪大学 大学院情報科学研究科 コンピュータサイエンス専攻

E-mail: [†]{m-itii,matusita,inoue}@ist.osaka-u.ac.jp

概要

ソフトウェア部品とはモジュールやクラス等のソフトウェアの構成単位であり，参照や呼び出しなどの形で互いに利用関係をもつ．ソフトウェア部品を頂点，利用関係を有向辺としたグラフはソフトウェア部品グラフまたは部品グラフと呼ばれ，ソフトウェアの解析手法に広く用いられている．グラフを特徴づける要素として頂点の次数分布がある．近年，*WWW* 上のページのリンク関係やソーシャルネットワーク等，様々な分野のグラフで次数分布がべき乗則に従うことが明らかになり，活発に研究されている．本研究では部品グラフの次数分布からソフトウェアの設計の特徴を得ることができないかどうか調査を行う．最近の研究により，ひとつのソフトウェアに基づく部品グラフの次数分布がべき乗則に従うことが報告されているが，個々のソフトウェアの性質と次数分布との関連については知られていない．本研究では，様々な部品集合に基づく部品グラフの次数分布を比較することにより，その関連について調査する．その結果，利用関係が特定の部品に集中する度合いなどの設計の特徴が次数分布に反映されることが明らかになった．また，次数分布に基づくソフトウェアの設計評価について考察する．

1　はじめに

近年，短期間で大規模かつ高品質なソフトウェアを開発するための技術に対する需要が高まっている．その中でも，ソフトウェアの解析に基づく手法はソフトウェアの評価や理解支援，再利用支援など様々な分野で見られる．ソフトウェアの解析における重要な概念としてソフトウェア部品グラフがある．モジュールや関数，クラスなどのソフトウェアの構成単位はソフトウェア部品（部品）と呼ばれ，互いに利用関係を持つ．ソフトウェア部品グラフ（部品グラフ）は，部品を頂点，部品間の利用関係を有効辺としたグラフのことを指す．

様々な分野において，グラフは研究対象の表現形式として用いられ，その性質が調査・研究されている．グラフを特徴づける要素のひとつに頂点の次数分布があり，それがべき乗則に従うグラフは特徴的な性質をもつことから物理学や社会学，生物学，情報科学など様々な分野で注目されている [1, 2, 9]．部品グラフに関しては，ひとつの Java または C++ソフトウェアに基づく部品グラフがべき乗則に従うことが報告されている [8, 12]．これらの研究では，いくつかのソフトウェアに共通して見られる特徴の調査が行われている．また，その共通する特徴に関して，設計や開発プロセスとの関連について考察されている．しかし，対象となる部品グラフはそれぞれひとつのソフトウェアに基づいたものであり，複数のソフトウェアを含む部品集合の部品グラフに対する調査は行われていない．また，対象ソフトウェアの設計による部品グラフの次数分布の違いについての議論はされていない．

この研究は，多数のソフトウェアシステムのデータ収集・分析を通じて，より良いソフトウェアシステムの構築を目指すという，Mega Software Engineering[5] の一種である．ソフトウェアはその用途や規模により様々なアーキテクチャやパターンを用いて設計される [3]．同じ機能をもつソフトウェアであっても，設計が異なれば構成する部品やその利用関係が異なる．これより，設計の異なるソフトウェアの部品グラフは，次数がべき乗則に従うという共通の性質をもちながらも，互いに異なる特徴をもつと考えられる．逆に，部品グラフの次数の特徴か

ら，もとのソフトウェアの設計の特徴を得て評価へつなげることが出来ると考えられる．

そこで，本研究ではひとつのソフトウェアだけでなく複数のソフトウェア集合に基づく部品グラフの次数分布について，べき乗則に従うかどうかという観点から調査する．このとき，部品集合による違いに注目して調査し，ソフトウェアの設計と次数分布との関連についての考察をおこなう．具体的には，部品グラフの次数分布の調査をおこない，ソフトウェア設計との関連を，次数の大きい部品の内容を確認し，次数とメトリクスとの相関を求めることにより調査する．このとき，部品間の利用関係において，利用することと利用されることは明らかに異なる意味をもつことから，入力次数と出力次数は区別して調査する．調査対象は，Java の基本ライブラリである JDK および複数のオープンソースソフトウェア，それらの集合からなるソフトウェア群とし，対象同士の結果を比較することにより，ソフトウェアによる違いや，ソフトウェア単体に利用ライブラリを加えた時の影響，および大規模なソフトウェア集合がもつ特徴について調査する．また，以上で得られた結果をもとに，ソフトウェアの設計と部品グラフの次数の分布との関連について考察する．

以降，2 節で関連研究について述べ，3 節でソフトウェア部品グラフなどの諸定義について述べる．4 節で実験について述べ，5 節で得られた結果に関して考察する．最後に 6 節で本研究のまとめと今後の課題について述べる．

2　関連研究

Valverde らは，JDK や Java アプリケーションのクラス図を，クラスおよびインターフェースを部品，汎化や依存といった関連辺を利用関係とする部品グラフとみなした時，次数に関してべき乗則が成り立つことを示している [12]．また，グラフ上の辺が頂点数に対して少数であるにも関わらず頂点間の最短距離が小さいスモールワールド性 [13] を持つことも示している．さらに，これらの性質は再利用性や理解容易性などを考慮し，最適なソフトウェア設計を行った結果であると考察している．

Myers は C++アプリケーションのクラス図を部品グラフとみなした時，次数がべき乗則に従うこと，またグラフがスモールワールド性をもつこと，および階層的な構造をもつことを示している [8]．Myers の実験では，次数は入力と出力に分けて調査され，入力次数と出力次数はいずれもべき乗則に従う点では共通しているが，それぞれ異なる性質を持つことが示されている．また，部品グラフと同様の性質をもつグラフの生成モデルを提案している． 以上の研究で用いられた部品グラフは，ソースコードなどから得た静的な利用関係に基づいている．

これに対し，Potanin らは動作する Java プログラムを解析し，オブジェクト間のメッセージのやりとりの関係がべき乗則に従うことを示している [11].

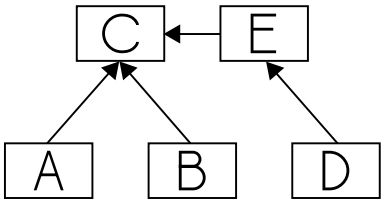

図 1. 部品グラフの例

3　準備

3.1　ソフトウェア部品グラフ

一般に，ソフトウェア部品は広義にはモジュールや関数，クラスなどのソフトウェアの構成要素のことを，狭義には再利用を目的として設計されたソフトウェアの実体のことを指す．本研究では広義のソフトウェア部品を用いる．以降，ソフトウェア部品を単に部品と呼ぶ．

ソフトウェアは構成要素の部品間で相互に属性や振る舞いを利用し合うことで 1 つの機能を提供する．いま，ある部品がある部品を利用する時，この部品間に利用関係が存在すると言う．

部品を頂点，利用関係を有向辺としたグラフを部品グラフと呼ぶ．部品グラフの例を図 1 に示す．図 1 では，部品 A, B, E は C を，D は E をそれぞれ利用していることを表している．

3.1.1 本研究における定義

本研究では，以下に定義する部品を頂点，部品間の利用関係を有向辺として部品グラフを構成する．

部品 Java クラスおよびインターフェース．

利用関係 以下の 6 通りの，Java ソースコードを静的に解析して得られる利用関係．

- **継承** 部品がクラスもしくはインターフェースを継承
- **実装** 部品がインターフェースを実装
- **変数宣言** 部品がクラスまたはインターフェースを変数として宣言．フィールドの型，メソッドの返値および引数の型として宣言した場合を含む．
- **生成** 部品がクラスのインスタンスを生成
- **メソッド呼び出し** 部品がクラスもしくはインターフェースのメソッドを呼出．呼び出されたメソッドが継承されたものである場合は，メソッド宣言の存在するクラスもしくはインターフェースに対しての利用関係となる．
- **フィールド参照** 部品がクラスもしくはインターフェースのフィールドを参照．参照されたフィールドが継承されたものである場合は，フィールド宣言の存在するクラスもしくはインターフェースに対しての利用関係となる．

3.2 べき乗則

本研究では，部品グラフの特徴として，入力次数および出力次数の分布がそれぞれべき乗則に従うかどうかに注目する．べき乗則は，Pareto の法則，Zipf の法則とも呼ばれ [10]，論文の共著関係 [9]，WWW 上のページのリンク関係 [1] など様々な分野において見られる．

ある分布がべき乗則に従うとき，ある要素 X が値 x をもつ確率 P は，x の $-a$ 乗に比例する．これを確率分布関数で表すと式 (1) となる．

$$P[X = x] \sim x^{-a} \tag{1}$$

この分布をプロットする時には一般に両対数軸が用いられる．通常の軸を用いてプロットすると図 2（a）の様になり，特徴を捉えづらいためである．

式 (1) の両辺の対数を取ると式 (2) のようになり，ある分布がべき乗則に従うときは，両対数軸でプロットした時に点が傾き $-a$ の直線上に並ぶことがわかる（図 2（b））

$$\log P[X = x] \sim -a \log x \tag{2}$$

しかし，実際のデータの度数をプロットすると，値の大きな部分にばらつきが生じるため，累積度数を用いてプロットする．式 (1) を累積分布関数で表すと式 (3) のようになり，この場合も両対数軸を用いてプロットすると値が直線上に並ぶことがわかる（図 2 （c））．

$$P[X > x] \sim x^{-(a-1)} \tag{3}$$

式 (3) の（すなわち，式 (1)，(2) の）a の値は，累積度数を両対数軸上にプロットした結果を直線へ線形回帰したときの直線の傾きから求められる．また，分布がべき乗則に従う度合いとして，回帰時の自由度調整済み寄与率 R^{*2} を用いる [6]． R^{*2} は，回帰モデルがデータを説明出来ている割合を示す値であり，0 から 1 の値をとる．

3.3 ソフトウェアメトリクス

ソフトウェアメトリクス（メトリクス）は，一般にはソフトウェアの開発活動中に計測された値から得られる特性値を意味するが，本研究では，個々の部品そのものから得られる部品の特性値のことを指す．実験において，以下に示す 4 種類のメトリクスと，入力および出力次数との相関を求めることにより，部品の性質と次数との関連を調査する．

LOC コメント行を除くソースコードの行数．

NOM クラスに定義されたメソッドの数．親クラスから継承したメソッドは含まない．

WMC 重みにサイクロマチック数を用いた，重み付きメソッド和 [4]．メソッド中の分岐や繰り返しに基づく値であり，この値が大きいほど複雑である．

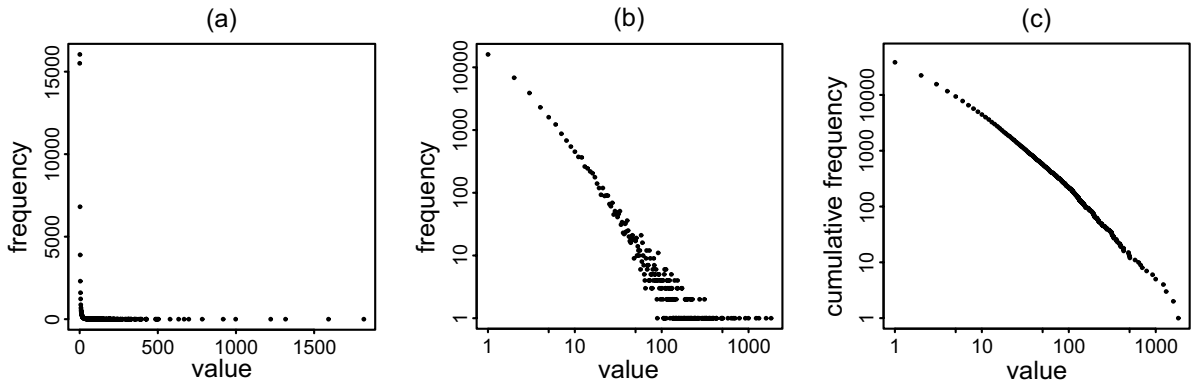

図 2. べき乗則のプロット

4 実験

4.1 部品集合

実験に用いる 6 種類の部品集合について，頂点数（部品数），辺数（利用関係数），総行数を表 1 に示す．またそれぞれの概要を以下に示す．

APACHE Apache Project[1]のリポジトリからのソフトウェア群．

JDK Java の基本ライブラリである JDK1.4

ECLIPSE オープンソースの Java 開発環境である Eclipse[2] ．

ECLIPSE+JDK Eclipse に，利用するライブラリである JDK を加えたもの．ただし，Eclipse から利用されない部品については除いてある．

NETBEANS Eclipse と同様にオープンソースの Java 開発環境である NetBeans[3]．

ALL Apache, Eclipse, NetBeans, SourceForge[4] など，様々なオープンソースソフトウェアのリポジトリからのソフトウェア群．

[1]http://www.apache.org/
[2]http://www.eclipse.org/
[3]http://www.netbeans.org/
[4]http://sourceforge.net/

4.2 実験内容

以下の 2 つの実験を行い，その結果に関する考察を行う．

実験 1 入力次数，および出力次数の分布の調査．
累積度数を対数軸を用いてプロットし，直線に回帰する事により式 (1) における a の値を求める．

実験 2 入力次数および出力次数の分布とソフトウェアに含まれる部品および設計との関連の調査．
部品の次数とソフトウェアメトリクスとの相関を求める．また，次数が上位の部品の内容をソースコードを基に調査する．

4.3 実験結果

実験 1 部品集合それぞれについて，部品グラフの入力次数の累積度数分布のプロットを図 3 に，出力次数の累積度数分布のプロットを図 4 に示す．プロットは両対数軸を用いている．また，それぞれのプロットについて，直線に回帰することで求めた式 (1) の a の値および R^{*2}

表 1. 実験で用いる部品集合

	頂点数	辺数	総行数
APACHE	59,486	303,775	4.2M
JDK	11,556	107,198	1.1M
ECLIPSE	13,941	140,678	1.3M
ECLIPSE+JDK	16,666	212,284	1.6M
NETBEANS	13,466	69,187	1M
ALL	180,637	1,808,982	14M

の値を表 2 に示す.

実験 2　部品集合 ALL について，部品の次数とソフトウェアメトリクスとの相関を求めた結果を表 3 に示す. また，それぞれの部品集合内にて，入力次数および出力次数が上位である部品（クラス）の中から特徴的なものを以下に示す.

APACHE 入力: ログを出力するクラスなど，複数のソフトウェアから共通して利用されるユーティリティクラス.

出力: オブジェクトのインスタンスをプールするクラスおよびバイトコード解析ツールの解析部に含まれるクラス．他の部品集合と比較して，上位部分に不自然に部品数の多い部分があらわれている．これは，その出力次数をもつ，ほぼ同内容のクラスが複数存在するためである.

JDK 入力: ALL と同様, String クラス，Object クラス.

出力: Java GUI フレームワークである AWT の実装クラス.

ECLIPSE 入力: 独自の GUI 実装である SWT のクラスや例外クラス，ソフトウェア内のリソースにアクセスするためのインターフェースクラス.

出力: Java エディタの実装クラスや，AST(抽象構文木) の実装クラス

ECLIPSE+JDK 入力: JDK と同様.

出力: ECLIPSE と同様.

NETBEANS 入力: ロケールごとの表示メッセージを扱うクラスや，IDE 内でのデータ構造を扱うクラス.

出力:　仮想ファイルシステムやエディタの実装クラス.

ALL 入力: 文字列を扱うクラス java.lang.String および すべてのクラスの親クラス java.lang.Object．それぞれ Java プログラム中でほぼ必ず利用されるクラスである．また，上位はほとんど JDK のクラスにより占められる.

出力: オブジェクトのインスタンスを プールするクラスおよびグラフ描画ツールのサンプルアプリケーション.

4.4　実験結果の考察

得られた結果に関して，部品集合間で共通する特徴および異なる特徴について考察する.

共通点　図 3 より，入力次数の分布はどれもべき乗則に従っていることがわかる．ALL, APACHE では入力次数の対数分布がほぼ直線に並ぶのに対して，単体ソフトウェアである JDK, ECLIPSE, NETBEANS では，値の大きい部分でやや傾きに変化が見られる．複数のソフトウェアの集合の場合，JDK などの共通して利用されるクラスへ辺が集中し，極端に大きな値をもちやすいためであると考えられる．JDK を含む部品集合 ALL, JDK, ECLIPSE+JDK では，JDK の中でも特に利用される Object および String が極端に大きな入力次数をもつことがわかる．また，このような部品集合では R^{*2} の値も高く，理想的なべき分布に近い．a の値については部品集合に関わらずほぼ 2 となり，部品集合の比較を行う時に一般的な値として利用できると考えられる.

図 4 より，出力次数については，値の大きい部分に関してはべき乗則に従っていると言えるが，次数 0 の付近で傾きに変化がみられる．この傾向は [8] では観測されていない．利用関係の取得をより詳細化した結果であると考えられる．この分布は Double-Pareto 分布であると考えられる [7]．この分布は値の小さな部分では対数正規分布，大きな部分ではべき分布であるような分布であり，ファイルサイズにみられる分布である．出力次数は部品内の記述内容から得られる値であり，また LOC との相関が見られたことからこの分布に従うと考えられる．直線に回帰して得た a の値は 3〜4 となり，多くのクラスを含む ALL, APACHE ではより大きな値となっている．傾きが大きいことは全体の部品数と比較して次数の大きな部品が少ないことを示しており，頂点数や辺数を大きくしても値域に大きな変化はないことを示している.

表 3. 次数とメトリクス値との相関

	出力次数	LOC	NOM	WMC
入力次数	0.004	0.071	0.239	0.075
出力次数	-	0.826	0.641	0.750

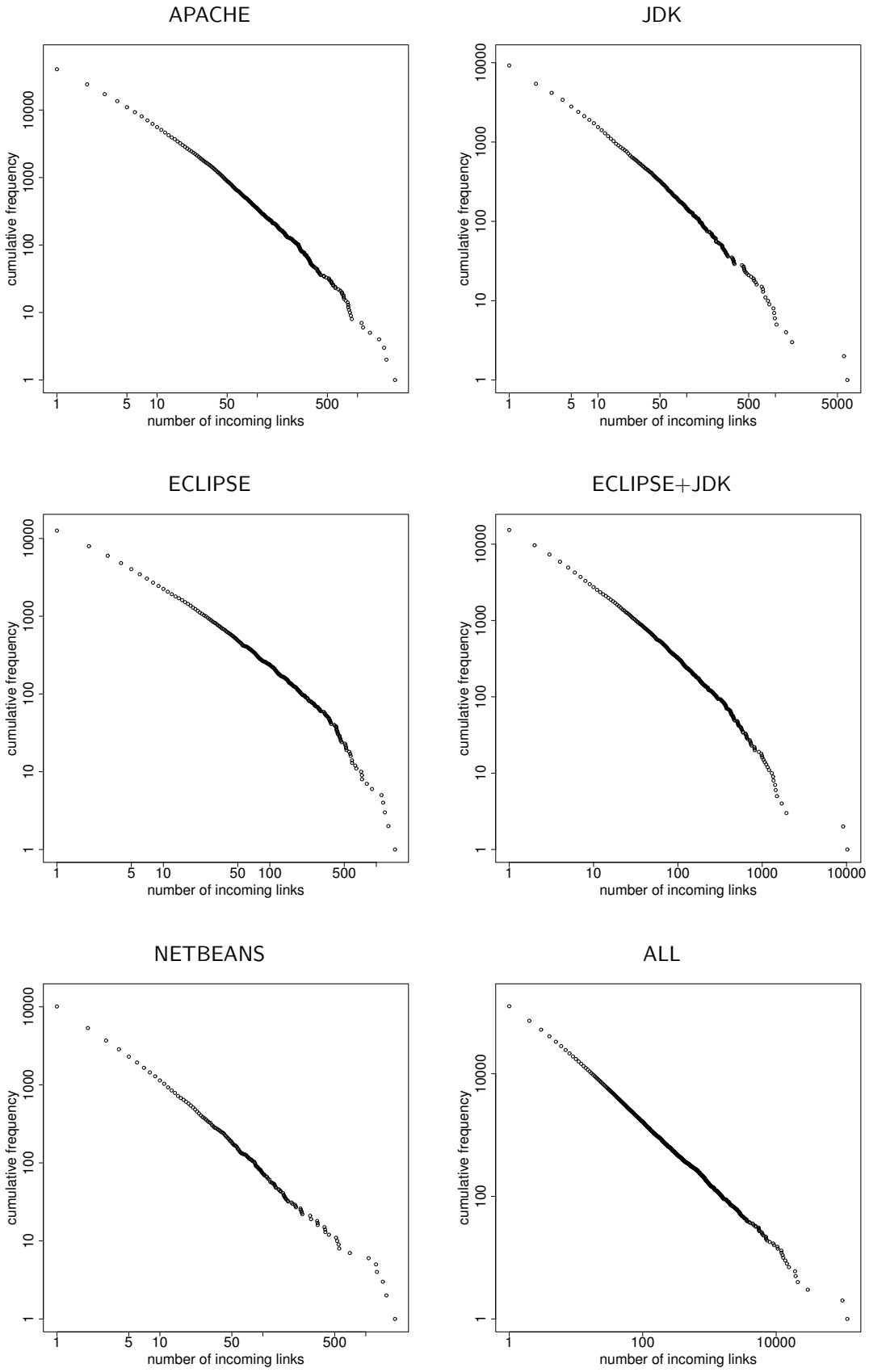

図 3. 入力次数の累積度数分布

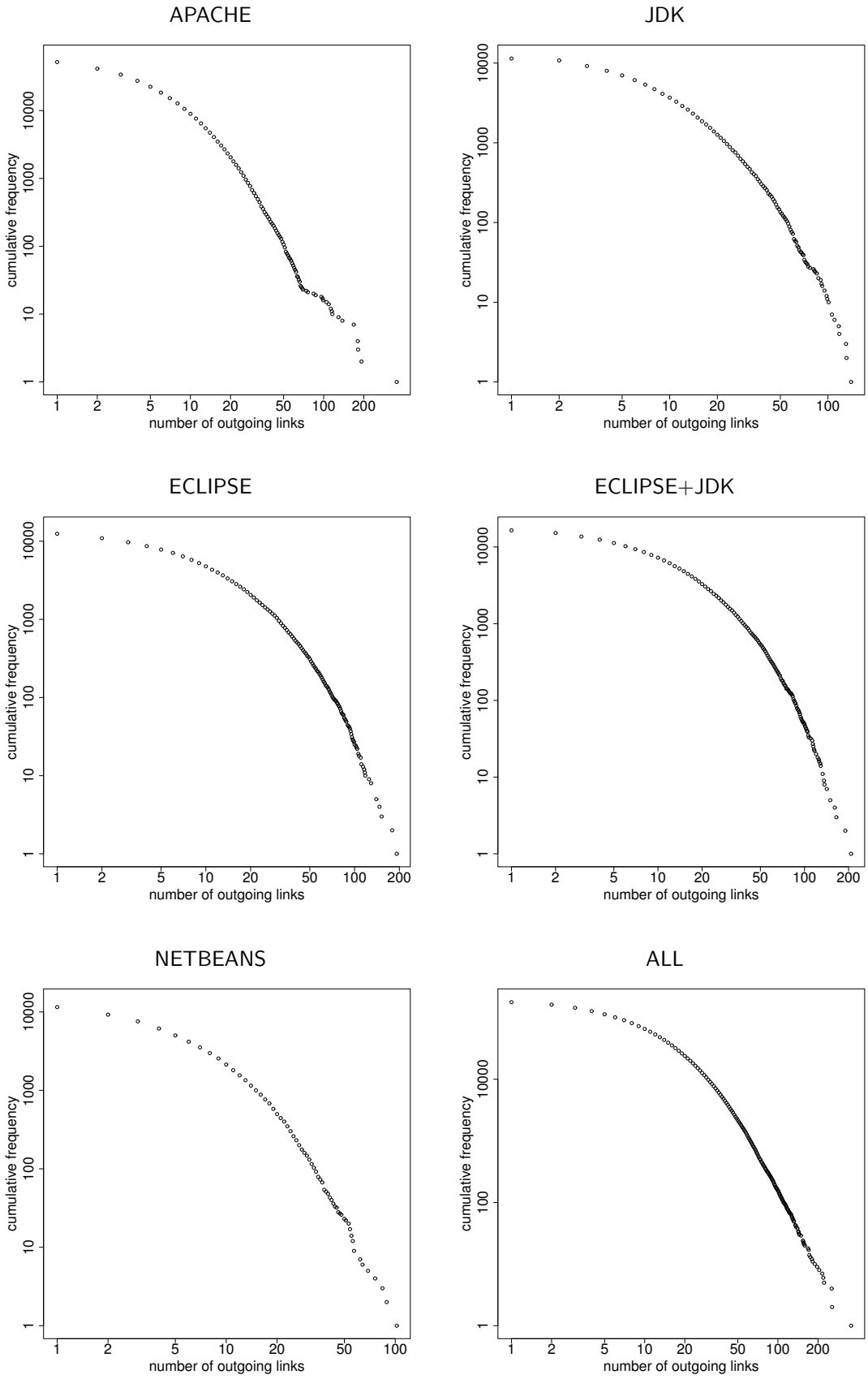

図 4. 出力次数の累積度数分布

	入力次数				出力次数			
	a			R^{*2}	a			R^{*2}
APACHE	2.37	±	0.0109	0.981	3.41	±	0.0637	0.942
JDK	2.11	±	0.00866	0.987	3.11	±	0.0818	0.876
ECLIPSE	2.20	±	0.0163	0.955	3.02	±	0.0770	0.856
ECLIPSE+JDK	2.11	±	0.0100	0.976	3.07	±	0.0790	0.844
NETBEANS	2.20	±	0.00788	0.994	3.32	±	0.1043	0.890
ALL	2.02	±	0.00145	0.999	3.66	±	0.0693	0.903

表 2. 回帰の結果

ソフトウェア集合同士の比較　APACHE と ALL を比較する．入力次数に関しては，どちらも高い R^{*2} をもち分布形状もほぼ直線となるが，ALL では横軸，つまり入力次数の最大値が 10000 以上であるのに対して，APACHE では最大 1000 程度と含まれるクラス数の差以上の差が出ている．これに対して，出力辺数に関しては共通点として述べた様に大きな違いは無い．APACHE に関しては値の大きな部分に段差の様な形状が見られるが，これはほぼ同じ内容のクラスが複数存在して同次数の要素が値の大きな範囲に複数存在することが原因である．

ひとつのソフトウェアとソフトウェアの集合の比較　ひとつのソフトウェアの部品集合 ECLIPSE, NETBEANS と複数のソフトウェアの集合である ALL, APACHE を比較すると，入力次数の分布に関して，プロット形状や，直線に回帰した際の R^{*2} の値から ALL および APACHE はより高い度合いでべき乗則に従う．これは，ソフトウェアの集合では，共通して利用されるクラスが JDK などの基礎的なクラスや XML パーサなどのアプリケーションのドメインに依存せず利用されるクラスに限られ，利用関係の集中が生じる為である．これに対してひとつのソフトウェアの場合は，入力次数のプロットの形状を見たときに値の大きな部分でカットオフと呼ばれる傾きの変化が生じている，つまり利用関係の集中の度合いが低い．この要因としては，母数の少なさの他に，レイヤー構造などの階層性をもつ設計がなされていることが挙げられる．よく設計されたソフトウェアであれば，基礎的な役割を持つ部品を直接扱う事を避け，ある程度の役割をもつ部品を作成し，その部品を利用する傾向がある．例えば，XML ファイルを設定ファイルとして用いているソフトウェアの場合，設定ファイルを扱う時に直接 XML ファイルを扱う部品を利用せず，設定ファイルを扱う為に作成した部品を利用することが多いと考えられる．

ソフトウェアがライブラリを含む場合と含まない場合の比較　ECLIPSE と ECLIPSE+JDK を比較する．入力次数については，ECLIPSE では上位のクラスが，ECLIPSE+JDK では JDK のクラスに押し下げられている．JDK のクラスは様々なソフトウェアで利用されるだけでなく，単体ソフトウェアの中でも多く利用されることがわかる．逆に，出力次数の上位は ECLIPSE と ECLIPSE+JDK で差が少ない．出力次数は，それぞれのクラスから利用されるクラスがどの程度集合に含まれるかということに依存するため，大部分を占める Eclipse のクラスが上位に来たと考えられる．また，JDK 単体では上位に来る AWT などのクラスを Eclipse が必要としていないことも要因の一つであると考えられる．

類似ソフトウェア間の比較　どちらも同程度の規模をもつ統合開発環境である ECLIPSE, NETBEANS をみる．入力次数に関する特徴として，どちらも上位付近で傾きが変化し，大きくなるという点が見られる．変化しはじめる点は，NETBEANS が約 1000 付近なのに対し，ECLIPSE は小さく約 500 である．このことは R^{*2} の値の差としてもあらわれており，NETBEANS と比較して ECLIPSE はやや低い値である．上位で傾きが大きくなることは，上位のクラスの次数も極端に大きくないことをを示す．また，最大値は NETBEANS の約 1900 に対して ECLIPSE は約 1500 であり，こちらも ECLIPSE の方が小さい．逆に，出力次数では分布の形状はほぼ同じであるのに対して，最大値は NETBEANS の約 100 に対して ECLIPSE は約 200 と大きな値をもつことから，全体として ECLIPSE の方が利用部品数が大きな傾向にある．

以上より，ECLIPSE の方が入力次数の最大値が小さく，比較的大きな値をもたない分布であることから，特定のクラスへの利用関係の集中度は低いと言える，また，そ

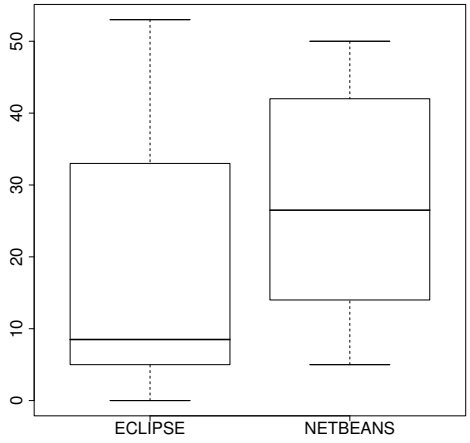

図 5. 入力次数上位部品の public メソッド数の比較

れぞれ総行数や行数の分布にそれほど差がないのに対し，出力次数では最大値で 2 倍程度という大きな差が見られることから，同じ行数のコードであっても ECLIPSE はより多くの部品を利用していることがわかる．これらのことから，NETBEANS と比較した ECLIPSE のクラス設計の特徴としては，それぞれのクラスに与えられている責任が細分化され，多くのクラスの相互作用で処理をおこなう点があると考えられる．実際，ECLIPSE の入力次数の上位には抽象クラスやインターフェースが多く，中心的なクラスは抽象度が高い役割をもっていることが分かる．また，入力次数上位のクラスの責任の度合いの比較するため，それぞれの public メソッド数の比較をおこなった．クラスの public メソッドは他のクラスに提供する機能であり，個数が多ければそのクラスが持つ役割や責任が大きいと考えられる．入力次数上位 10 クラスの public メソッド数の一覧を表 4 に，Boxplot により比較した結果を図 5 に示す．これより，ECLIPSE の方が public メソッド数が少ない傾向がわかる．このことからも，ECLIPSE の入力次数上位のクラスは NETBEANS と比較して細分化された役割や責任をもっていることがわかる．

5　考察

実験より，入力次数，出力次数はいずれも同様の形状となる，つまり同様の分布に従うが，詳細に見ていくと互いに異なる点をもつことが明らかになった．その違いは，含まれる部品やソフトウェア自体の設計を反映していると考えられる．このことから，次数の分布から，ソフトウェアの設計に関して大局的な評価をすることが出来ると考えられる．

まず，入力次数の分布がべき乗則に従う度合いが高いソフトウェアは，設計が不適切である可能性がある．実験により，入力次数の分布はいずれもべき乗則に従うが，部品集合がひとつのソフトウェアの場合にはカットオフが見られた．4.4 節で述べたとおり，これはレイヤー構造などの適切な設計をもつことが原因だと考えられる．逆に，カットオフが見られず，べき乗則に従う度合いがより高い場合には適切に設計されていないために利用関係が極端に集中していることが考えられる．ただし，この観点から評価を行うためには基準や目安となる値が必要であり，今後調査する必要がある．

また，入力次数と出力次数から，中心的なクラス設計の役割の細分化の度合いを測ることができると考えられる．ECLIPSE と NETBEANS の比較により得られた通り，入力次数の分布についてカットオフの度合いが大きく，出力次数の最大値が大きい時，中心的なクラスの役割が細分化されている傾向がある．また，その逆も言える．このときに極端な結果となる場合には何らかの問題が存在する可能性がある．

表 4. 入力次数上位部品の public メソッド数の比較

入力次数の順位	public メソッド数	
	ECLIPSE	NETBEANS
1	6	21
2	5	39
3	8	49
4	53	14
5	9	11
6	33	42
7	0	32
8	43	5
9	29	50
10	5	14
MEAN	19.1	27.7
MEDIAN	8.5	26.5

6 まとめ

本研究では，ひとつのソフトウェアや複数のソフトウェア集合に基づく部品グラフの入力次数および出力次数について調査した．その結果，いずれも入力次数はべき乗則に従い，出力次数はべき乗則に類似した分布に従うことが明らかになった．また，それぞれを比較することにより，利用関係が特定の部品に集中する度合いなどの設計の特徴が次数分布に反映されることが明らかになった．また，次数分布に基づくソフトウェアの設計評価についての考察を行った．

今後の課題としては，次数分布に基づきソフトウェア設計の評価を行うため，より多くの部品集合や，異なる利用関係に基づく部品グラフを対象として実験を行い，ソフトウェアの設計と次数分布のより詳細な関連について調査することが挙げられる．

謝辞

本研究の一部は文部科学省「e-Society 基盤ソフトウェアの総合開発」の委託に基づいて行われた．また，本研究の一部は，文部科学省 21 世紀 COE プログラム（研究拠点形成費補助金）及び文部科学省科学研究費補助金萌芽研究（課題番号: 18650006）の研究助成を受けている．ここに記して謝意を表す．

参考文献

[1] R. Albert, H. Jeong, and A.-L. Barabási. Diameter of the world-wide web. *Nature*, 401(6749):130–131, September 1999.

[2] A.-L. Barabási. *Linked: The New Science of Networks*. NHK 出版, 2002.

[3] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. *Pattern-Oriented Software Architecture: A System of Patterns*. John Wiley & Sons, 1996.

[4] S. R. Chidamber and C. F. Kemerer. A metrics suite for object oriented design. *IEEE Transactions on Software Engineering*, 20(6):476–493, June 1994.

[5] K. Inoue, P. K. Garg, H. Iida, K. Matsumoto, and K. Torii. Mega software engineering. In *In Proc. of the 6th International Conference on Product Focused Software Process Improvement(PROFES'05)*, pp. 399–413, June 2005.

[6] 久米, 飯塚. 入門 統計的方法 2 回帰分析. 岩波書店, 1987.

[7] M. Mitzenmacher. Dynamic models for file sizes and double pareto distributions. *Internet Mathematics*, 1(3):305–333, 2003.

[8] C. R. Myers. Software systems as complex networks: Structure, function, and evolvability of software collaboration graphs. *Physical Review E*, 68(4):046116, 2003.

[9] M. E. J. Newman. The structure of scientific collaboration networks. In *In Proc. of the National Academy of Sciences of USA 98*, pp. 409–415, 2001.

[10] M. E. J. Newman. Power laws, pareto distributions and zipf's law. *Contemporary Physics*, 46:323–351, 2005.

[11] A. Potanin, J. Noble, and M. Frean. Scale-free geometry in oo programs. *Communications of the ACM*, 48(5):99–103, May 2005.

[12] S. Valverde, R. Ferrer-Cancho, and R. V. Solé. Scale-free networks from optimal design. *Europhysics Letters*, 60(4):512–517, 2002.

[13] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, June 1998.