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

信学技報
TECHNICAL REPORT OF IEICE.

ソフトウェア部品の利用関係におけるスケールフリー性の調査

市井　誠[†]　松下　誠[†]　井上　克郎[†]

† 大阪大学 大学院情報科学研究科
E-mail: †{m-itii,matusita,inoue}@ist.osaka-u.ac.jp

あらまし　スケールフリー性とはグラフ中のノードの接続辺数がべき分布に従う性質であり，インターネット上のノードなど様々な対象において確認されている．ソフトウェア部品間の利用関係をあらわす部品グラフにおいても，大部分の部品はほとんど利用関係を持たないのに対してごく少数の部品が非常に多くの利用関係を持つスケールフリー性をもつことが知られている．利用関係はソフトウェアの設計が反映されたものであることから，部品グラフにおける接続辺数の分布から設計に関する情報が得られると考えられるが，設計による接続辺数の分布の違いは知られていない．本研究では，ソフトウェアによる接続辺数の分布の違いを入力辺数と出力辺数に分けて調査をおこない，ソフトウェアおよび含まれる部品の性質との関連について調査する．その結果，入力辺数および出力辺数の分布はソフトウェアの設計や含まれる部品により異なることが判明した．また，得られた結果より，理解支援やソフトウェア評価を目的として，部品グラフの接続辺数の分布からソフトウェアの設計に関する特徴の分析をおこなう手法について考察する．
キーワード　ソフトウェア部品，利用関係，部品グラフ，スケールフリー

Exploration of Scale-freeness of Use-relations of Software Components

Makoto ICHII[†], Makoto MATSUSHITA[†], and Katsuro INOUE[†]

† Graduate School of Information Science and Technology, Osaka University
E-mail: †{m-itii,matusita,inoue}@ist.osaka-u.ac.jp

Abstract　Scale-freeness is a graph property that distribution of number of edges which are connected to vertices are power-law and appears in various targets including component graphs which represents relationships between software components. Components graphs reflect software design and we think that we can get useful information about software design by analyzing the distribution. In this paper, we explore distribution of number of incoming and outgoing edges respectively and relationship with software design. As a result, we found that distributions reflects softwares to another. We also discusses analyzing method of software design.
Key words　Software component, Use-relation, Component graph, Scale-free

1. はじめに

ソフトウェアは複数の部品から構成され，それぞれの部品の間には利用関係が存在する．ソフトウェアは，部品を頂点，利用関係を有向辺としたグラフとして考えることができる．部品グラフは，ソフトウェアの理解や評価などを目的とした様々な手法に利用されており，その性質を知ることは重要である．

ソフトウェアの部品グラフにおいて知られている性質として，スケールフリー性がある [6] [8]．グラフにおけるスケールフリー性とは，グラフ上の各頂点に接続される辺の数がべき分布に従う，すなわちある接続辺数 n をもつ頂点の個数が n の負のべき乗に比例する性質のことをいう．スケールフリー性は様々な分野においてみられ，例えば WWW 上のページ間のリンクやインターネット上のノードなどが挙げられる．インターネット上のノードの場合，スケールフリー性をもつことで，ランダムな障害に強く，狙いすました攻撃に弱いという特徴をもつことが知られている [2].

これまでのソフトウェアにおけるスケールフリー性に関する研究では，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おこなわれている．また，その共通する特徴に関し，設計ポリシーや開発プロセスとの関連についての考察がされている．しかし，単体の中小規模のソフトウェアが対象であり，また，対象ソフトウェア設計による分布の違いについては議論されていない．

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，

スケールフリー性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，部品グラフの接続辺数の特徴から，もとのソフトウェアの設計の特徴をつかむことが出来ると考えられる．

そこで，本研究ではソフトウェアの部品グラフの接続辺数の分布を対象ソフトウェアによる違いに注目して調査し，対象ソフトウェアの設計と分布との関連についての考察をおこなう．具体的には，部品グラフの接続辺数の分布の調査をおこない，その分布ともとのソフトウェアの関連を，接続辺数の大きい部品の内容を確認することおよび接続辺数とその他の部品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，入力辺数と出力辺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．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接続辺数の分布との関連について考察する．また，部品グラフの接続辺数の分布をもとにソフトウェアの設計の情報を得て，ソフトウェア理解支援をおこなう手法についての考察をおこなう．

また，上記に加え，手続き指向言語である C 言語で記述された，FreeBSD のカーネルおよびユーザランドのソースコードを対象とした調査も行う．これまでの研究は主としてオブジェクト指向言語を対象としているが，手続き指向言語で記述されたソフトウェアでも成立するかどうか調査する．

以降，2. 節で本研究で対象とする部品グラフについて述べ，3. 節でスケールフリー性について述べる．4. 節で実験内容および結果について述べ，5. 節で結果に関する考察をおこなう．6. 節では C 言語のソフトウェアに対する調査について述べる．最後に 7. 節でまとめと今後の課題について述べる．

2. 部品グラフ

一般にソフトウェア部品とは再利用を前提に設計された部品のことをさすが，ここではより一般的に，ソフトウェアを構成する単位をさしてソフトウェア部品，または単に部品と呼ぶ．

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一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．図 1 は部品グラフで，この場合，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．

本研究では Java ソフトウェアを対象としており，クラスおよびインターフェースが部品となる．また，利用関係は Java 部品間の任意の利用関係を扱う．すなわちクラスの継承・インターフェースの実装・変数宣言・インスタンス生成・メソッド呼び出し・フィールド参照を利用関係として扱う．なお，利用関係は Java ソースコードの静的解析により得る．

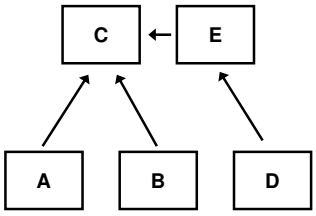

図 1　部品グラフ

3. スケールフリー性

グラフ上の頂点の接続辺の数がべき分布に従うとき，グラフにスケールフリー性があるという．べき分布は下式の確率分布関数であらわされる [1].

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

式 (1) は，ある要素 X が値 x をもつ確率 P は，x の $-a$ 乗に比例することをあらわしている．ある分布がべき分布に従うことは，横軸を値，縦軸をその値を持つ要素の個数として両対数軸でプロットした場合に直線上に並ぶことにより確認される．このときの直線の傾きは式 (1) における a となる．ただし，現実のデータをプロットした場合，値の大きな部分でばらつきが生じるため，縦軸を，ある値以上をもつ要素の個数としてプロットし，直線に並ぶかどうかを見ればよい．これは，式 (1) を累積分布関数で表すと下式のようになることによる．

$$P[X > x] \sim x^{-(a-1)} \tag{2}$$

3.1 ソフトウェアにおけるスケールフリー性

これまでに，ソフトウェアに関するスケールフリー性に関していくつかの研究がなされている. [8] では， JDK 等のクラス図を無向グラフとして扱ったとき，スケールフリー性をもつこと，およびスモールワールド性 [2] をもつことを示している．グラフには階層性があり，柔軟性や再利用性，複雑性など，トレードオフを含む様々な要素を考慮してクラス設計をおこなうことにより，このようなグラフが形成されると考察している．また, [6] では，ソフトウェアの部品グラフが，スケールフリー性・スモールワールド性をもつことを，有向グラフであることを考慮しつつ示している．また，このような特徴をもつ部品グラフを生成する，ソフトウェア開発をシミュレートするモデルを提案している．

これらは静的な部品グラフを対象としているのに対して, [7] では動作する Java プログラム中でのオブジェクト間の利用関係がスケールフリー性をもつことを示している．

本研究では，ソフトウェア部品間の静的な利用関係をもとに部品グラフを構成する．また [8] [6] においては複数のソフトウェアに共通する性質に焦点をおいているのに対し，対象ソフトウェアによる違いに焦点をおいた調査をおこなう．

4. 実　　験

4.1 実験内容

ソフトウェアの部品グラフにおけるスケールフリー性を調査するため，以下の実験を行う

	D1	D2	D3	D4	D5	D6
入力辺数	2.02	2.37	2.11	2.20	2.12	1.99
出力辺数	3.66	3.4	3.12	3.02	3.08	3.21

表 1 a の 値

4.1.1 ソフトウェアの解析方法

Java のソースコードに対し，SPARS-J [4] を用いて静的解析をおこない，部品グラフやメトリクス値を得る．対象とするデータセットは以下の D1〜D6 の 6 種類である．

D1 Apache Project(注1)，SourceForge(注2)，NetBeans(注3)，Eclipse(注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．(約 18 万クラス，1400 万行)

D2 Apache Project からのソフトウェア群． (約 6 万クラス，450 万行)

D3 Java の基本ライブラリである JDK1.4 (約 11000 クラス，110 万行)

D4 オープンソースの Java 開発環境である Eclipse (約 14000 クラス，130 万行)

D5 D4 に，JDK などの利用ライブラリを加えたもの．ただし，利用ライブラリに含まれるクラスのうち，利用されないものは除いてある． (約 17000 クラス，160 万行)

D6 Eclipse と同様にオープンソースの Java 開発環境である NetBeans．(約 13000 クラス，100 万行)

4.1.2 実 験 手 順

実験は，まず (1) 入出力辺数の分布の調査をおこない，続いて (2) 分布とソフトウェアの関連の調査をおこなう．(2) では，値が上位の部品の内容をソースコードにより確認するとともに，部品の入出力辺数といくつかの他のメトリクスとの比較をおこなう．この時に比較するメトリクスは，LOC(コメントを除いた行数(注5)) よび NOM(メソッド数) である．

4.2 実 験 結 果

4.2.1 実験 (1): 入出力辺数の分布

データセット D1〜D6 に関して，入力辺数の分布を図 2 に，出力辺数の分布を図 3 に示す．

それぞれの図の横軸は入力辺数もしくは出力辺数であり，縦軸は部品数である．図での軸はすべて両対数軸でとってあり，累積度数でプロットしている．

また，式 (2) での a の値，すなわち両対数軸にプロットした値に対してひいた回帰直線の傾きを表 1 に示す．

4.2.2 実験 (2): 分布とソフトウェアの関連

a） 部品内容の確認

データセットごとに，入力辺数/出力辺数の上位数クラスに対し，内容を確認した結果を以下に示す．

D1 入力辺数の上位: 極端に大きなクラスが存在するが，それらは java.lang.String および java.lang.Object である．それぞれ文字列を扱うクラス，全てのクラスの親クラスであり，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．

出力辺数の上位: 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のサンプルアプリケーション．

D2 入力辺数の上位: ログ出力クラス他，共通して利用されるユーティリティクラス．

出力辺数の上位: D1 でも上位であったプールクラスおよびバイトコード解析ツールの解析部．D1 などと比較し，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，その出力辺数をもつ，ほぼ同内容のクラスが複数存在するためである．

D3 入力辺数の上位: D1 と同様，String クラス，Object クラス．

出力辺数の上位: Java GUI フレームワークである AWT の実装クラス．

D4 入力辺数の上位: 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

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

D5 入力辺数の上位: D3 と同様，String や Object をはじめ JDK のクラス．

出力辺数の上位: D4 と同様

D6 入力辺数の上位: 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

出力辺数の上位: 仮想ファイルシステムやエディタの実装クラス

b） メトリクス値との比較

データセット D1, D4 に対し，入出力辺数とメトリクス値との順位相関係数を求めたものを表 2 に示す．

これより，入力辺数は他のメトリクスとほとんど相関が無いことがわかる．また，出力辺数は，行数およびメソッド数とも相関が高い．行数やメソッド数は，それぞれ異なる意味で部品の規模をあらわしており，出力辺数も，同様に部品の規模をあらわしていると言える．

5. 考 察

5.1 実験結果の考察

5.1.1 共 通 点

実験 (1) で得られた結果より，入力辺はどれもべき分布に従っており，スケールフリー性をもつといえる．D1, D2 では入力辺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 D3, D4, D5 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からであると考えられる．JDK を含むデータセット D1, D3, D5 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辺数をもつ．また，傾きについてはデータセットに関わらずほぼ 2 となり，データセットの比較の際に一般的な値として利用できると考えられる．

出力辺数については，値の大きい部分に関してはべき分布に従い，スケールフリー性を示していると言えるが，0 付近で傾

(注1)：http://www.apache.org/
(注2)：http://sourceforge.net/
(注3)：http://www.netbeans.org/
(注4)：http://www.eclipse.org/
(注5)：ここでの定義．一般に LOC はコメントを含む行数を指す．

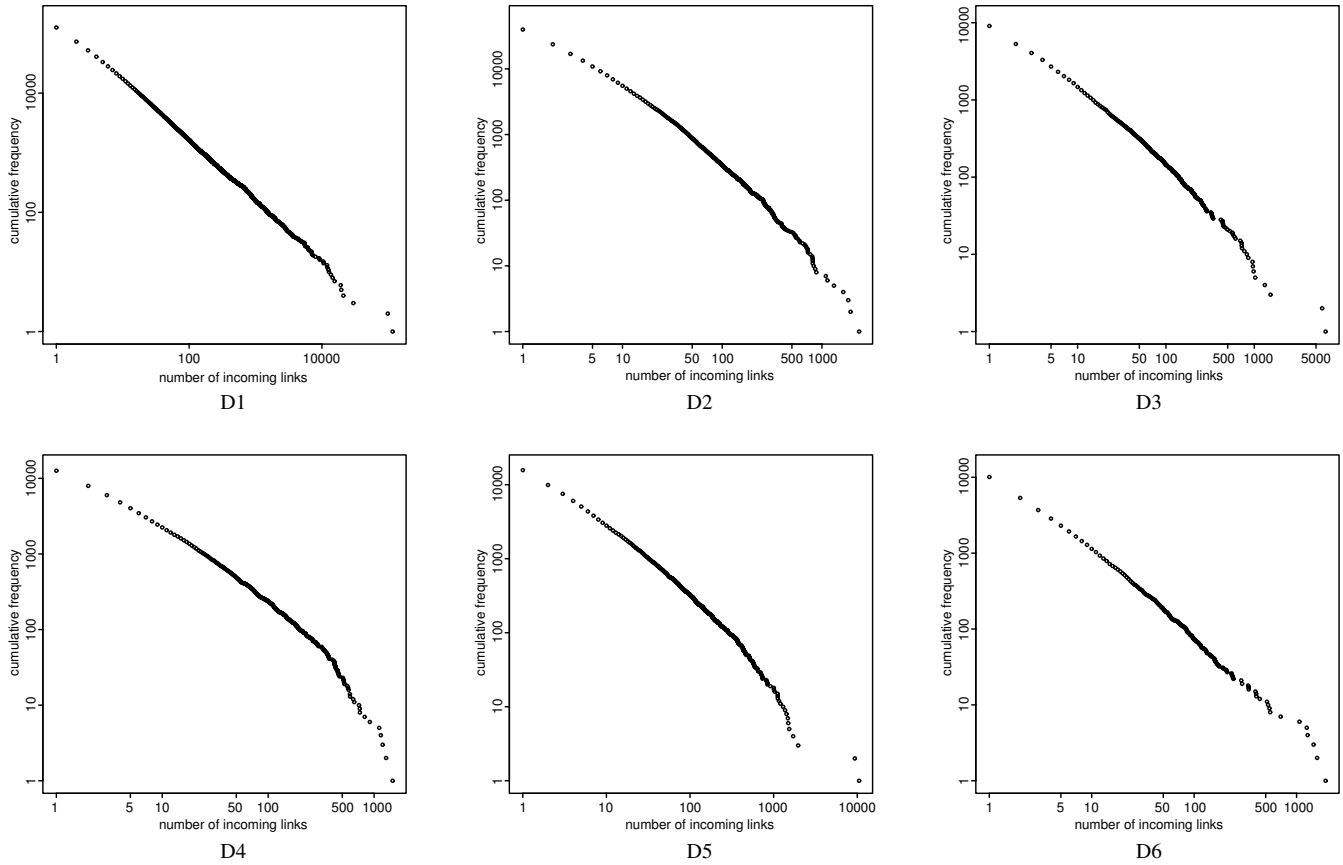

図 2　入力辺数の累積度数分布

	出力辺数	LOC	NOM
入力辺数	0.0044	0.071	0.24
出力辺数	-	0.83	0.64

D1

	出力辺数	LOC	NOM
入力辺数	-0.081	-0.055	0.22
出力辺数	-	0.73	0.63

D4

表 2　メトリクス値との相関

きに変化がみられる．この分布は，0 付近では対数正規分布に従う Double-Pareto 分布 [5] であると考えられる．この分布はファイルサイズに見られる分布であり，部品内の記述内容から得た出力辺数がこの分布に従うことは妥当であると考えられる．また，回帰直線の傾きは 3〜4 となり，多くのクラスを含む D1, D2 ではより大きな値となっていることから，データセットに含まれるクラス数と関連があると考えられる．

5.1.2　データセット同士の比較

得られた結果を各データセット間で比較し含まれるソフトウェアやソフトウェア部品との関連について考察する．まず多数のソフトウェアを含むデータセットである D1 と D2 を比較し，続いて単体のソフトウェアと，それに JDK などの利用するライブラリまで含めた D4 と D5，最後にいずれも Java 開発環境であり，ほぼ同様の規模である D4 と D6 について比較をおこなう．

D1 と D2　入力辺数に関しては，どちらもほぼ直線となるが，D1 では横軸，つまり入力辺数の最大値が 10000 以上であるのに対して，D2 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．入力辺数の分布はデータセットの要素により大きく異なり，スケールフリー性の文字通りの意味である「特定のスケールを持たない」という性質を示すことがわかる．しかし，出力辺数に関しては，それぞれのデータセット単体では分布の形状からスケールフリー性をもつようにみえるが，互いに比較した時には横軸のスケールに大きな差は見られない．このことから，出力辺数はスケールフリーに近い性質をもちながらも何らかの上限をもつことがわかる．これは，出力辺数の多いクラス，すなわち多くのクラスへの利用関係を持つクラスは必然的にそれ自身が巨大となり，再利用性や保守性などで問題がおきるために現実には限度があることを反映していると考えられる．

D4 と D5　入力辺数については，D4 では上位のクラスが，D5 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，出力辺数の上位には D4 と D5 で差が少ない．出力辺数は，部品中で利用されるクラスがどの程度データセット中に含まれるかということに依存するため，データセット中で多くを占める 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．

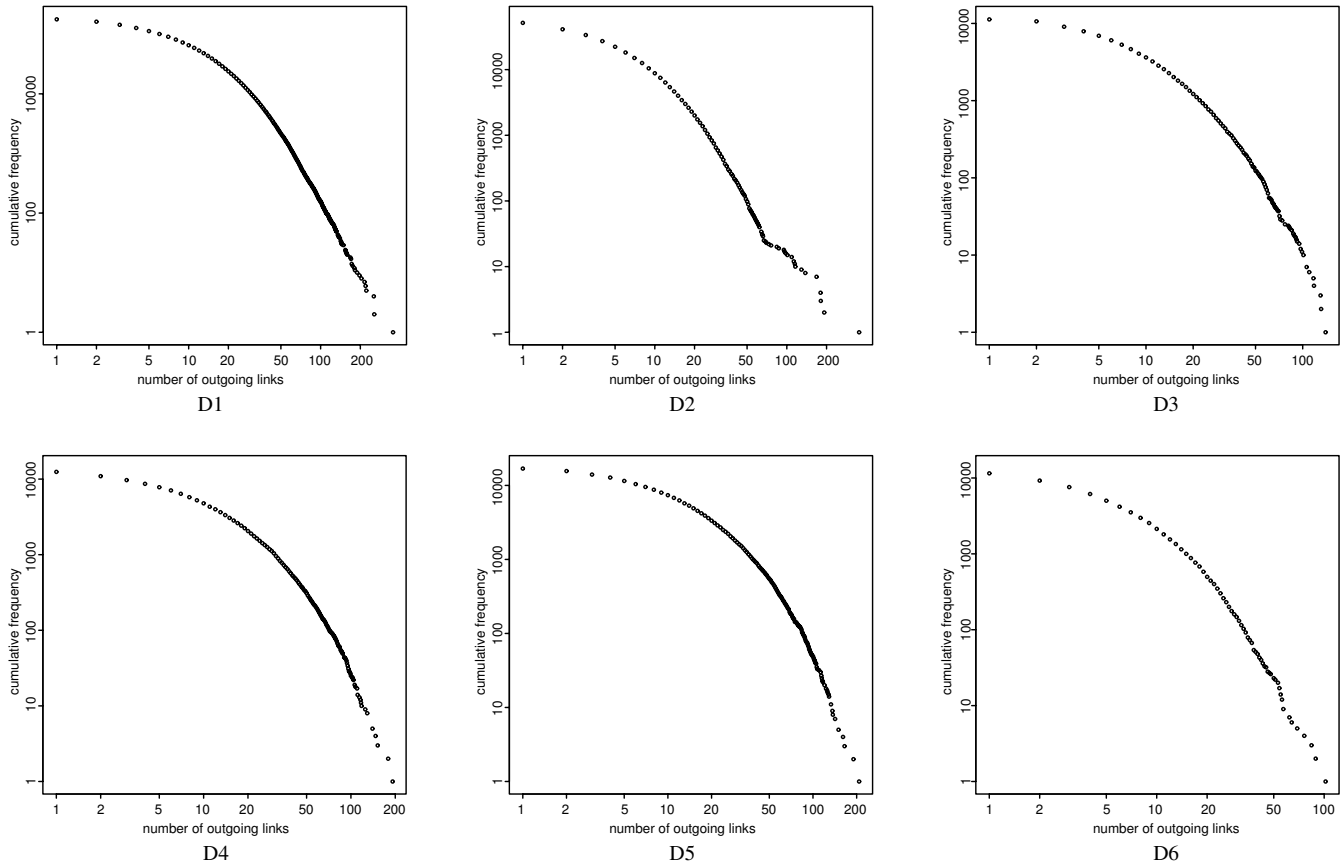

図 3　出力辺数の累積度数分布

D4 と D6　入力辺数に関して，どちらも上位付近で傾きが変化し，大きくなるという特徴がある．変化しはじめる点は，D6 が約 1000 付近なのに対し，D4 は小さく約 500 である．上位で傾きが大きくなることは，上位のクラスも極端に大きな値は持たないことを示す．また，最大値は D6 の約 1900 に対して D4 の約 1500 と，こちらも D4 の方が小さい．逆に，出力辺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 D6 の約 100 に対して D4 の約 200 と大きな値をもつことから，全体として D4 の方が利用部品数が大きな傾向にあるといえる．

以上より，D4 の方が入力辺の最大値が小さく，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，出力辺数では最大値で 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 D4 はより多くの部品を利用することがわかる．これらのことから，D6 と比較した D4 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，D4 の入力辺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．

5.2　応用に関する考察

以上を踏まえ，部品グラフの入力辺数および出力辺数の分布の，クラス設計面からのソフトウェア理解や評価への応用について考察する．

まずソフトウェア理解への応用について考察する．ソフトウェアのクラス設計の理解には中心的なクラスを知る必要があり，Component Rank(CR) を利用した手法を提案した [9]．提案した手法では，ソフトウェア中で中心的な役割を持つクラスは高い CR 値を持つことに注目し，まずソフトウェア中のクラスの CR を求め，CR 順に全クラスをソートしたうえで，その上位よりクラスの役割および関連を理解する作業をおこなう．少数の重要なクラスから開始することで，ソフトウェアを効率よく理解することができる．しかし，理解の単位となる上位クラスの個数に関して具体的な個数は示せていない．ソフトウェアにより中心的なクラスの個数にばらつきがあり，例えば「全クラス数の 1 割」などのように示せないためである．

本研究により，入力辺の分布，および出力辺の分布は基本的にスケールフリー性をもつが，ソフトウェアのクラス設計，とくに中心的に使われるクラスの抽象度などの設計を反映した特徴をもつことがわかった．これより，中心的なクラスの個数についても部品グラフと関連があり，部品グラフの入出力辺数の分布から得られると考えられる．現段階では具体的な関連については不明であるが，今後，部品グラフの入出力辺数の分布と中心のクラスの個数の関連を調査することで，CR を利用した理解支援手法において，上位のクラスの個数を自動的に決定することができると考えられる．

またソフトウェア設計の評価の面からは，出力辺や入力辺の

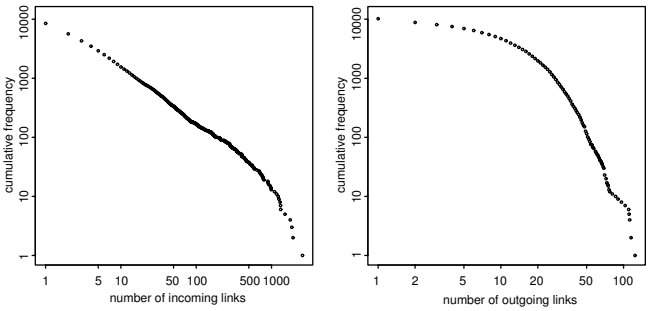

図 4　FreeBSD のソースコードの利用関係の分布

分布をもとに，ソフトウェアの設計の妥当性について観点から評価できると考えられる．入力辺の分布から特定のクラスへの依存度の高さを，出力辺および行数の分布から複雑さを評価することができると考えられる．これらに関して一般的な値を求めることが出来れば，特定のソフトウェアに関して，値が大きく外れている場合には設計に問題があるとしてリファクタリングを推奨する設計評価手法を提案できると考えられる．

6. C 言語に対する調査

ここまで，オブジェクト指向言語である Java で記述されたソフトウェアを扱ってきた．これまでの研究では，Java や C++ といったオブジェクト指向言語で記述されたソフトウェアが主として扱われてきたが，手続き指向言語で記述されたソフトウェアでも同様になりたつかどうか調査する．

ここでは，C 言語で記述された FreeBSD5.4 のカーネルおよびユーザランドのビルド時に生成されるオブジェクトファイルを対象として調査した．部品グラフの頂点は 1 つのオブジェクトファイル，辺は関数・グローバル変数などのシンボルである．頂点数は約 11000 となった．

4. 節と同様にプロットした結果を図 4 に示す．これより，Java ソフトウェアと同様に，入力辺数は上位付近で傾きが変わるべき分布，出力辺数は Double-Pareto に近い分布となっている．入力辺数の上位はメモリ領域やファイルポインタなどを扱うものなど基本ライブラリである libc の一部で，出力辺数の上位は認証を扱うモジュールやファイルシステムの修復をおこなうコマンドなど様々な機能を提供するものである．

一般のオブジェクト指向ソフトウェアでは抽象度の高い概念が入力辺数の上位であるのに対して，FreeBSD は低レベルな操作を扱うものが上位である．他のファイルがこれらを利用して OS の様々な機能を実現することにより，スケールフリー性があらわれていると考えられる．入力辺数の上位にて傾きが変化している点については，オブジェクト指向ソフトウェアと異なり，関数でラップするなどして間接的に利用した場合に利用関係が引かれないことも理由の一つであると考えられる．

7. まとめと今後の課題

本研究では，対象データセットによる部品グラフの各頂点の入力辺数および出力辺数の分布の違い，および，データセットに含まれるソフトウェアとの関連についての調査をおこなった．その結果，どのデータセットについてもスケールフリー性という点では共通しているが，共通ライブラリの有無や，ソフトウェア設計により分布に違いが発生することが明らかになった．また，部品グラフの入力辺数および出力辺数の分布をもとに，ソフトウェア理解支援や評価をおこなう手法についての考察をおこなった．

今後の課題としては，CR について，その分布や，接続辺数との関連の調査が挙げられる．CR は部品グラフに基づく部品の順位付け手法であり，入力辺数との関連があるのは明らかであるため，CR での順位と入力辺数の順位が大きく異なる部品に関しては，何らかの設計的要因が関連していると考えられ，その要因を調査することで，クラス設計の分析に役立てることができると考えられる．

また，5. 節で述べた理解支援や評価への応用に関しての調査が挙げられる．まず，理解支援手法の段で述べた，ソフトウェア内で中心となる部品の数と，入力辺数および出力辺数の分布の関連の調査である．これには，様々なソフトウェアに対し，まず中心となる部品を確定させた上で，分布を調査する必要がある．評価の手法の為の調査としては，まず出力辺数や入力辺数の分布を数値化することが挙げられる．最大値や，べき分布の指数の他に「傾きが途中で変化する」といった分布そのものに関する情報を数値として得られる必要がある．そして，一般的なソフトウェアでの値・および問題となる値を調査する必要がある．

最後に，調査結果に基づき手法を提案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) の研究助成によるものである．ここに記して謝意を表す．

文　献

[1] L. A. Adamic: ”Zipf, Power-laws, and Pareto - a ranking tutorial”, http://www.hpl.hp.com/research/idl/papers/ranking/ranking.html (2000)

[2] A. Barabasi: ”Linked: The New Science of Networks”, Perseus Books Group (2002)

[3] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal: ”Pattern-Oriented Software Architecture, Volume 1: A System of Patterns ”, John Wiley & Sons (1996)

[4] K. Inoue, R. Yokomori, T. Yamamoto, M. Matsushita, S. Kusumoto: ”Ranking Significance of Software Components Based on Use Relations”, *IEEE Transactions on Software Engineering*, Vol.31, No.3, pp.213-225 (2005)

[5] M. Mitzenmacher ”Dynamic Models for File Sizes and Double Pareto Distributions”, *Internet Mathematics*, vol.1, no.3, pp.305-333 (2003)

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

[7] A. Potanin, J. Noble and M. Frean: ”Scale-free Geometry in OO Programs”, *Communications of the ACM* (2005)

[8] S. Valverde and R. V. Sole: ”Hierarchical Small Worlds in Software Architecture”, http://www.santafe.edu/research/publications/wpabstract/200307044 (2003)

[9] 市井, 横森, 松下, 井上: ” コンポーネントランクを用いたソフトウェアのクラス設計に関する分析手法の提案”, 電子情報通信学会技術研究報告, SS2005-37, Vol.105, No.229, pp.25-30 (2005)